E.T.S.I.S. TELECOMUNICACIÓN

**PROYECTO FIN DE CARRERA**
**PLAN 2000**

**TEMA:** Internet de las cosas IoT

**TÍTULO:** Sistemas de Gestión de sensores mediante Tecnologías M2M

**AUTOR:** Miguel de Miguel Heredero

**TUTOR:** Rafael Herradón Diez **Vº Bº.**

**DEPARTAMENTO:** DIAC

**Miembros del Tribunal Calificador:**

**PRESIDENTE:** Eduardo Juarez Martínez

**VOCAL:** Rafael Herradón Diez

**VOCAL SECRETARIO:** Florentino Jimenez Muñoz

**DIRECTOR:**

**Fecha de lectura:** 16 de Junio 2014

**Calificación:** **El Secretario,**

**RESUMEN DEL PROYECTO:**

En el proyecto se lleva acabo un estudio practico sobre dos escenarios donde intervienen dispositivos relacionados con el Internet de las cosas.

El primer escenario consta de la configuración y montaje de un microcontrolador avanzado conocido como Waspmote que se encargar de recoger variables atmosféricas gracias a un conjunto de sensores y envía los datos a un router multiprotocolo Meshlium mediante tecnología Zigbee, un tipo de red orientada a redes de sensores.

La segunda parte dos dispositivos de hardware libre como son un Arduino con capacidad GPRS y una RaspberryPI conectada a la red cableada enviaran datos a una red social de sensores conocida como Xively, es una plataforma M2M gratuita que permite una gestión de flujos de datos y consultas de estos por otros medios. He diseñado una aplicación Android que permite la consulta de datos y administración de sensores por un usuario.

Se ha detallado las configuraciones y el proceso de instalación de todos los dispositivos.

En el proyecto se lleva a cabo un estudio práctico sobre dos escenarios donde intervienen dispositivos relacionados con el Internet de las cosas. También se puede situar como una solución de comunicación M2M. Comunicación máquina a máquina implica un sistema central que es capaz de conectarse con otros sistemas en varios lugares. La conexión permite que el sistema central recoja o envíe datos a cada lugar remoto para su procesamiento.

El primer escenario consta de la configuración y montaje de un microcontrolador conocido como Waspmote que se encarga de recoger variables atmosféricas gracias a un conjunto de sensores y enviar los datos a un router multiprotocolo Meshlium mediante tecnología Zigbee, un tipo de red orientada a redes de sensores. Este montaje tiene como fin instalar una estación meteorológica en el campus de la universidad y poder almacenar y administrar sus datos.

La segunda parte dos dispositivos de hardware libre como son un Arduino con capacidad GPRS y una RaspberryPi conectada a la red cableada enviaran datos por ejemplo de temperatura y luminosidad a una red social de sensores conocida como Xively, gestionaremos nuestros dispositivos sobre esta plataforma gratuita, que nos permite dar de alta dispositivos, almacenar y representar los datos en tiempo real y consultarlos vía Web o mediante una aplicación móvil realizada para este caso por medio de funciones ofrecidas por Xively. He diseñado una aplicación Android que permite la consulta de datos y administración de sensores por un usuario, intenta abstraer al usuario de la complejidad técnica y acercar los objetos conectados, en este caso sensores.

Se han detallado las configuraciones y el proceso de instalación de todos los dispositivos. Se explican conceptos para entender las tecnologías de comunicación, Zigbee y Http, este protocolo participara a nivel de aplicación realizando peticiones o enviando datos, administrando la capacidad y por tanto ahorro.

The project takes a practical study on two scenarios which involved related to the Internet of Things devices. It can also be placed as a M2M communication solution. Machine to machine communication involves a central system that is able to connect with other systems in several places. The connection allows the central system to collect or send data to each remote location for processing.

The first stage consists of the configuration and setup of a microcontroller known as Waspmote which is responsible to collect atmospheric variables by a set of sensors and send the data to a multiprotocol router Meshlium by Zigbee technology, a type of sensor networks oriented network. This assembly aims to set up a weather station on the campus of the university and to store and manage their data.

The second part two devices free hardware like Arduino with GPRS capacity and RaspberryPi connected to the wired network send data, temperature and luminosity to a social network of sensors known as Xively, manage our devices on this free platform, which allows us to register devices, store and display data in real time and consult the web or through a mobile application on this case by means of functions offered by Xively. I have designed an Android application that allows data consultation and management of sensors by a user, the user tries to abstract the technical complexity and bring the connected objects, in this case sensors.

Were detailed settings and the installation of all devices. Concepts are explained to understand communication technologies, Zigbee and Http, this protocol participate performing application-level requests or sending data, managing capacity and therefore savings.

A mis padres Antonio y Susana, a mi hermano Diego, gracias por vuestra paciencia y apoyo.

"*Ningún sensor mide el cariño y comprensión que he recibido estos años*"

# Contenido

# 1.INTRODUCCION

## 1.1.Motivación del proyecto

La idea de partida para el desarrollo del proyecto se basaba en el diseño de una plataforma de control de la temperatura por medio de una aplicación móvil. La intención desde un principio era llegar a programar nuestra propia aplicación Android, a través de la cual pudiésemos llegar a controlar a distancia la plataforma Arduino, que a su vez, se encargaría de la gestión de una red de sensores y actuadores capaces de detectar la temperatura ambiente en todo momento. La universidad adquirió un router Meshlium y una Waspmote así que los incluimos en el proyecto junto con las RaspberryPi para completar el grupo de dispositivos más utilizados para llevar acabo no solo prototipos sino despliegues reales, además de poder añadir al proyecto una red Zigbee, muy utilizada en redes de sensores.

Disponía de conocimientos previos con Arduino y estaba interesado en aprender a programar aplicaciones para dispositivos móviles. Realizar un proyecto donde pudiera participar y desarrollar a distintos niveles de abstracción, programación, electrónica y redes, además de realizar todas las partes de manera OpenSource (libre de licencia) motivó mi compromiso con los objetivos. El trabajar con temas de hardware y software libre facilita el aprendizaje debido a las distintas comunidades de usuarios que se ayudan mediante tutoriales y distintas guías, lo que permite ir superando los distintos obstáculos.

Existe una revolución respecto a la conexión de dispositivos, en el último año han surgido diferentes empresas que permiten la autogestión de sensores (flujos de datos) mediante un portal web, son conocidas como plataformas M2M, una de ellas Xively que comenzó promocionándose como una red social de sensores. Nos permite conectar nuestros dispositivos y ver representados en graficas el valor de nuestros diferentes sensores.

La mayoría de las empresas líder en el sector de la comunicaciones ofrecen sus soluciones tanto físicas como lógicas por tanto el proyecto además de formarme permite enfocarme a futuras salidas profesionales dado el auge del internet de las cosas o soluciones M2M, conceptos que se explicaran continuación.

La eficiencia energética se está viendo beneficiada al poder obtener datos en tiempo real, en el último año hemos recibido la llegada de contadores inteligentes en el hogar, no solo monitorizan si no que agilizan la gestión de datos por parte de las empresas eléctricas.

## 1.2.Planteamiento y objetivos del proyecto

Tras adquirir la universidad nuevos dispositivos el proyecto creció y nos permitió explorar distintos formas de gestionar sensores. Dos escenarios:

Una estación meteorológica (Fig. 1.1), formada por un router Meshlium y una Waspmote que tendrá conectada todos los sensores. Entre ellos se comunicaran mediante una red Zigbee.

Un Meshlium es un router que permite gestionar datos recibidos por interfaces con distintas tecnologías, GPRS, WIFI, GPS y para este caso Zigbee, el dispositivo nos permite desplegar una red de este tipo, el Meshlium toma un papel de nodo coordinador, recibiendo datos de una Waspmote.

La Waspmote (motas) son microcontroladores de bajo consumo, programables que facilitan la conexión y gestión de sensores, pueden comunicarse con otros nodos compartiendo información.

En el capítulo 2 explicare que características que tiene una red Zigbee, detallaremos las configuraciones necesarias de los distintos elementos para que la Waspmote establezca conexión con el coordinador, lea datos de los sensores, y cree una trama de datos interpretable por el Meshlium, tras recibir esté la trama, el router reconoce y almacena los datos por separado de los diferentes sensores, presión atmosférica, humedad, velocidad del viento, ect.

Meshlium es capaz de administrar redes de varios elementos y comunicarse con otros como él, pudiéndose utilizar para aplicaciones mayores. La solución tiene como fin establecer una estación de medida para las variables atmosféricas en el campus Sur. La Waspmote puede instalarse en el tejado junto con los sensores y mandar los datos al router que se encuentra en el laboratorio.

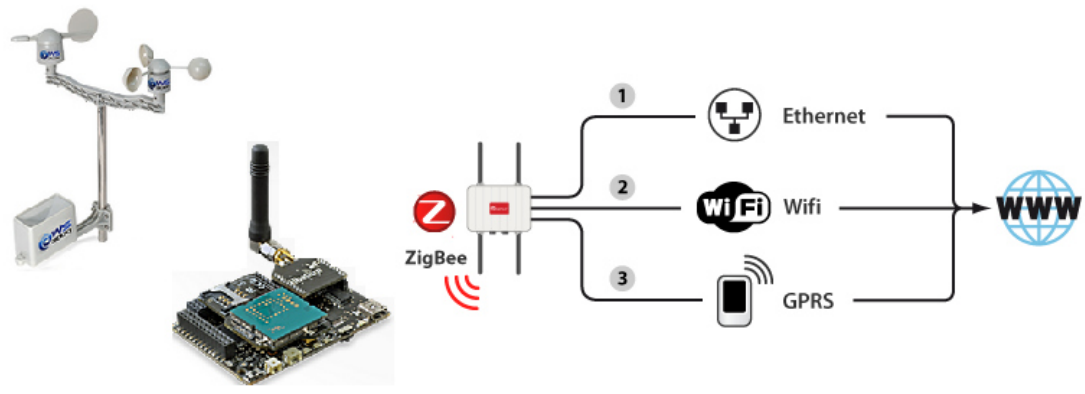



Fig. 1.1 Esquema general del escenario 1

Otro escenario de gestión de sensores:

Se enviara datos de varios sensores a una plataforma M2M y nosotros accederemos a los datos mediante una aplicación Android (Fig. 1.2).

En este apartado gestionaremos nuestros dispositivos a través de una plataforma gratuita, Xively, que nos permite dar de alta dispositivos, almacenar y representar los datos en tiempo real y consultarlos vía Web o mediante una aplicación Android realizada para este caso por medio de funciones que se comunican con Xively.

Los sensores estarán conectados a un Arduino Mega (otro microcontrolador), mediante GPRS enviara datos por medio de la red de comunicaciones móviles y la RaspberryPi hará lo mismo conectada a la red cableada, este último es un ordenador de bajo consumo, del tamaño de una tarjeta de crédito que permite conectar y leer sensores mediante un puerto de propósito general (GPIO).

En el capítulo 3 se detalla el uso de HTTP, protocolo de aplicación muy útil para mandar y recibir un pequeños conjuntos de datos, la configuración y particularidades de cada elemento hardware, explicare que es una plataforma M2M, como darse de alta y como funciona.

La aplicación Android para la consulta de datos se beneficia de funciones que provee Xively a los desarrolladores, abstrayendo al usuario de complejidades técnicas y facilita la interacción con los sensores. Por ejemplo la petición de ubicación del sensor que nos permitirá situar en un mapa de Google su localización.

Desarrollar una Aplicación móvil para usuarios sin capacidades técnicas planteo retos de usabilidad y accesibilidad.

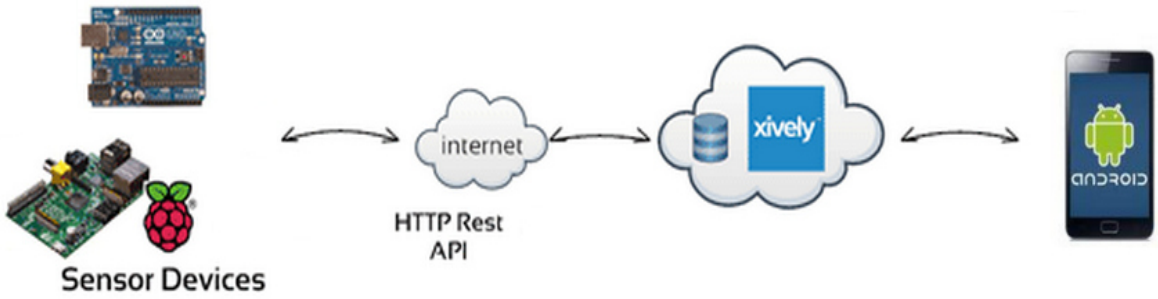



Fig. 1.2 Esquema general del escenario 2

Objetivos:

A nivel docente, Aprender cómo se despliega y configura una red de sensores, configurar una red Zigbee que permita la comunicación inalámbrica y centralizar el tráfico de datos en un único nodo con el fin de gestionar y administrar la información.

A nivel práctico, no es un prototipo con fines académicos. Se quiere establecer una estación meteorológica (primer escenario) en el campus que permita conocer en tiempo real las distintas variables atmosféricas. Para ello se disponen de varios dispositivos inteligentes, que encajan en el movimiento de Smartcities.

A nivel personal, enfrentarme a dispositivos y conceptos no adquiridos durante la carrera con aptitudes ya entrenadas, electrónica, programación y documentación.

En las siguientes paginas presento los dispositivos, detallo como si fuese un manual las configuraciones y explico de manera sencilla los diversos pasos llevados a cabo. Al ser diferentes dispositivos se han utilizado varios lenguajes de programación, lenguaje C, python, JAVA (Android), de estas partes solo se presentaran decisiones de diseño más relevantes y un diagrama de flujo que oriente del comportamiento para cada conjunto de código.

# 2.ESTADO DEL ARTE

## 2.1.Internet de las cosas (IoT)

Es un concepto difícil de definir con precisión, describe un mundo en el que casi cualquier cosa se puede conectar y comunicarse de una manera inteligente. En otras palabras, con el Internet de las cosas, el mundo físico se está convirtiendo en un sistema de información y de computación que describe un futuro en el que se conectarán los objetos físicos cotidianos a Internet y serán capaces de identificarse con otros dispositivos, generando todos estos un volumen de datos enorme.

Se estima que el volumen de dispositivos conectado a internet superara el de 40 billones para el 2020 (Fig. 2.1), para ese año la cantidad de datos será tres veces mayor.

Obviamente el futuro del IoT traerá tanto a empresas, instituciones como personas, nuevas formas de conexión. Por ejemplo las empresas cambiarán las relaciones con sus clientes, mejorará el ciclo de negocio y reducirá costos de operación.

El IoT y el análisis de datos generado por las personas serán de gran utilidad para descubrir tendencias y mejorar la sociedad en diversos campos. Ayudar a gobiernos a solucionar problemas, mejora de la seguridad ciudadana, prever y actuar ante emergencias y catástrofes, disminuir accidentes, mejorar la eficiencia energética etc.

Las propias empresas que quieran controlar sus datos sin subcontratar empresas externas, tendrán un problema para poder gestionar, administrar, almacenar y proteger toda esa información.

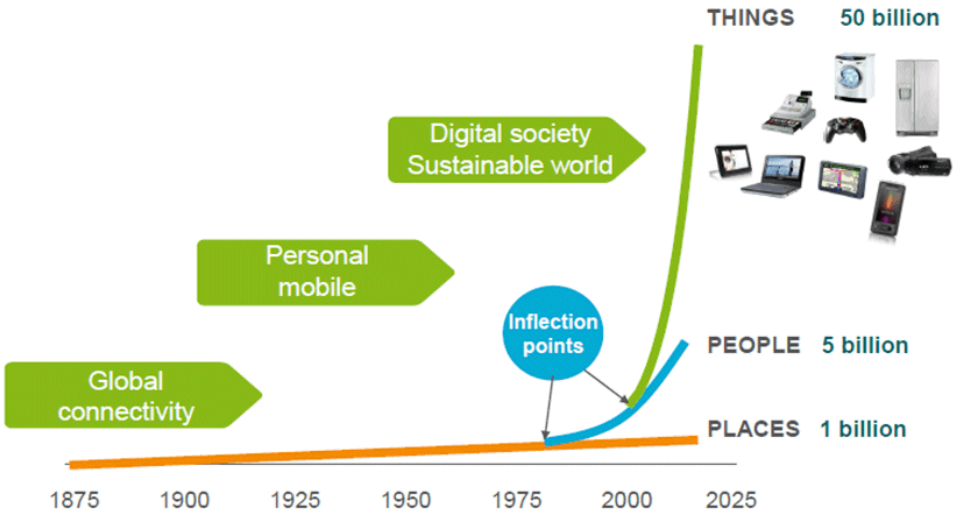



Fig. 2.1 Demanda de dispositivos conectados a internet

Cuestiones que surgen del IoT:

Internet de las cosas inmediatamente dispara preguntas en torno a la privacidad de los datos personales. Si la información en tiempo real acerca de nuestra ubicación física, o actualizaciones de la presión arterial, pueden ser accesibles por nuestros profesionales de la salud, que tienen nuevos tipos de datos más detallados acerca de nosotros.

Suministrar alimentación a esta nueva proliferación de dispositivos y sus conexiones de red puede ser costoso y logísticamente difícil. Todos estos elementos portátiles requieren baterías que algún día deben ser reemplazados.

Aunque muchos dispositivos móviles están optimizados para reducir el uso de energía, los costos de energía para mantener potencialmente miles de millones de ellos siguen siendo altos.

Numerosas corporaciones y empresas de nueva creación se han aferrado a la Internet de las Cosas, en busca de tomar ventaja de cualquier negocio y oportunidades disponibles.

En IoT se supone que el equipo de red subyacente y la tecnología conexa pueden operar semi-inteligente y, a menudo de forma automática. Basta con mantener los dispositivos móviles conectados a Internet. La gente tiene diversas necesidades que requieren un sistema IO adaptable y configurable para muchas situaciones y preferencias diferentes. Por último, a pesar de todos estos retos a superar, si la sociedad se vuelve demasiado dependiente de la automatización y la tecnología y no es muy robusto, cualquier fallo técnico en el sistema puede causar daños físicos o financieros.

Actualmente la IoT abarca miles de millones de objetos con capacidad de grabar, enviar y recibir datos automáticamente, como por ejemplo las pulseras deportivas para registro de actividad física, este tipo de dispositivos se clasifican como *Wearable*, su ubicación es nuestro cuerpo. Aquí incluiríamos las GoogleGlass y los relojes inteligentes (*smartwatches*), estos elementos afectaran nuestro estilo de vida, entretenimiento, salud y bienestar. Se puede considerar como una industria incipiente en la que participan las grandes empresas nos lo del sector de las comunicaciones, también multinacionales orientadas a productos deportivos o de ocio.

Se han producido cambios necesarios para tal cantidad de nuevos dispositivos conectados. IPv6 supone un factor clave en el desarrollo del concepto IoT. Gracias a este nuevo protocolo (diseñado para reemplazar a IPv4) se evitará que el crecimiento de Internet quede restringido, y hará posible la gestión de direccionamiento de innumerables dispositivos.

Por otro lado, ya son muchas las aplicaciones móviles y servicios en la nube que nos permiten la conexión a todos estos dispositivos, y proporcionan una vía para el tratamiento de una inmensa cantidad de datos en tiempo real (sistemas 'Big Data'), facilitando la integración de infinidad de sensores aplicables prácticamente a cualquier tipo de necesidad

## 2.2.Machine to Machine (M2M)

Máquina a Máquina (M2M) es una etiqueta amplia que se puede utilizar para describir cualquier tecnología que permite a los dispositivos conectados en red para intercambiar información y realizar acciones sin la asistencia manual de los seres humanos.

Es la utilización de diferentes tipos de dispositivos mecánicos para establecer una comunicación e intercambio de información. El M2M permite a las empresas controlar y manipular equipos de manera remota que es crucial para la operación del negocio. Este tipo de monitoreo y control remoto hace posible que las empresas puedan hacer frente a los problemas de servicio y restaurar la funcionalidad con poco tiempo y no afectar a la productividad.

La estructura básica de la tecnología M2M implica un sistema central que es capaz de conectarse con otros sistemas en varios lugares. La conexión permite que el sistema central recoja o envíe datos a cada lugar remoto para su procesamiento. Por ejemplo, la tecnología M2M se está utilizando para medir los contadores de electricidad en los hogares, estos dispositivos conocidos como *Smartmeters* permiten a las empresas eléctricas saber en todo momento y gestionar la demanda de energía en tiempo real, además de controlar el consumo de los usuarios.

Los componentes clave de un sistema M2M incluyen sensores, RFID, Wifi o conexión de comunicaciones móviles y software programado para ayudar a un dispositivo conectado

en red a interpretar los datos y tomar decisiones. El tipo más conocido de la comunicación M2M es la telemetría, que ha sido utilizada desde principios del siglo pasado para transmitir datos operacionales. Pioneros en la telemetría utilizaron por primera vez las líneas telefónicas y más tarde, en las ondas de radio para transmitir las mediciones de rendimiento obtenidos de instrumentos de seguimiento en lugares remotos.

Los estándares de Internet y la mejora de la tecnología inalámbrica han ampliado el papel de la telemetría de la ciencia pura, la ingeniería y la fabricación hasta el uso diario en productos como las unidades de calefacción, medidores eléctricos y aparatos conectados a Internet. Los productos fabricados con capacidades de comunicación M2M a menudo se comercializan a los usuarios finales como "inteligente".

Los escenarios más demandados son, el *vending*, las maquinas expendedoras de productos son ideales para estos usos, permitiendo saber las existencias de la maquina, el dinero recaudado, si la maquina está operativa o incluso comprar con un SMS. Medicina, muchos enfermos pueden desarrollar una vida casi normal y enviar a su médico de forma regular y automática los datos medidos en su propio domicilio evitando que tengan que desplazarse al hospital. Automoción, alarmas de automóvil que facilitan la ubicación del vehículo o monitorización de parámetros críticos del vehículo. Control de acceso, alarmas y demás sistemas para controlar el acceso a domicilios y empresas. Control y supervisión del trafico paneles informativos, semáforos, contadores de aforo de vehículos, sensores meteorológicos, Gestión de flotas ya no solo es posible la geolocalización de los vehículos además es posible conocer hasta los tiempos de descanso de los conductores y las veces que se abre el camión para controlar la carga. En el medio ambiente encontramos estaciones meteorológicas, niveles de agua en pantanos, energía solar y eólica. En definitiva miles de servicios que nos evitan desplazamientos innecesarios con el consiguiente ahorro energético que suponen por tanto la protección del medio ambiente.

Nuevos estándares como *Bluetooth Low Energy* (BLE) tecnología de bajo consumo de energía, permite a los nuevos dispositivos inteligentes puedan funcionar durante meses o años, baterías de tipo botón. BLE especialmente adecuado es para sensores, actuadores y otros dispositivos pequeños establece el consumo de energía extremadamente bajo muy útil para soluciones domóticas. Otras redes inalámbricas como el Zigbee nos permiten la configuración de los elementos de la red con diferentes topologías y jerarquías. Para el control de flotas resulta más útil la instalación de dispositivos con SIM y GPS para su geolocalización.

Las soluciones M2M abarcan desde los hogares donde encontramos electrodomésticos (frigorífico inteligente, aspiradora autónoma) que se comunican entre sí para no acaparar los recursos y ser más eficientes. A escenarios, como hospitales inteligentes, ciudades inteligentes (smartcities), con su control de tráfico, sistemas de aparcamiento y alumbrado monitorizado.

Los nuevos gobiernos deberán prestar atención a la creciente ocupación del espectro y la simplificación de estándares de la tecnología.

Para mantener la regulación de mercado, esta readecuación de los estándares tecnológicos debe hacerse de manera paralela con otros países. De este modo, internet pueda ser aprovechado a nivel global.

El hardware libre debido a su bajo coste unido a una comunidad que comparte conocimientos de manera desinteresada permite a los desarrolladores implementar ideas y casos de uso de sus aplicaciones, generando conocimiento y valor, se usan a menudo para hacer las primeras versiones de nuevos productos y prototipos para luego hacer el hardware definitivo miniaturizado.

A día de hoy es amplia la gama de dispositivos junto con los diferentes módulos de comunicación se ajustan a cada solución. En este proyecto usaremos diferentes dispositivos de Libelium, empresa española líder en el sector M2M/IoT.

# 3.ESCENARIO: ESTACION METEOROLOGICA

## 3.1.Introducción

La Waspmote(1) tiene que ser capaz de comunicarse con el Meshlium(2) mediante una red Zigbee. Esto dos elementos forman parte de un ecosistema desarrollado por Libelium para facilitar que empresas, o personas realicen sus soluciones y proyectos.


3.1 Nodos principales, Waspmote y el Meshlium

Nuestro Meshlium dispone de un interfaz Zigbee y la Waspmote de un modulo Xbee que nos permite conectarlos entre sí, pero como veremos tenemos que llevar a cabo una configuración para que ambos elementos se entiendan y establezcan una comunicación coherente.

Para este primer caso, comenzaremos definiendo que es una red Zigbee, características principales, pila de protocolos, topología, tipos de nodos que forman la red y los diferentes módulos de conexión que nos ofrece el mercado. Definir los distintos elementos que participan en el escenario, como se instalan los sensores y se configura la Waspmote tanto a nivel físico como software. El Meshlium se instala y configuraran todos los interfaces a modo de estudio. Y se detallara el proceso de arranque de ambos nodos detallando su comportamiento.

Se presentaran algunos elementos de la programación de la Waspmote y justificación de las diferentes decisiones, tanto para el envió de datos como para establecer modos de funcionamiento más eficientes.

## 3.2.Red Zigbee

### 3.2.1.¿Qué es una red Zigbee?

Cuando comencé a tratar este tema me producían confusión distintos términos que espero aclarar para que se entienda bien donde nos estamos moviendo.

IEEE 802.15.4 es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos.

Zigbee es desarrollado por la Zigbee Alliance (3), formada por varias compañías que quieren solventar la necesidad de un estándar para comunicaciones a baja velocidad, con un bajo coste de implementación y donde los dispositivos que forman parte de una red pueden requerir un bajo consumo. Se basa en el estándar de comunicaciones 802.15.4, es decir Zigbee implementa por encima de la capa MAC, esto es lo que se conoce como *Zigbee Stack*.

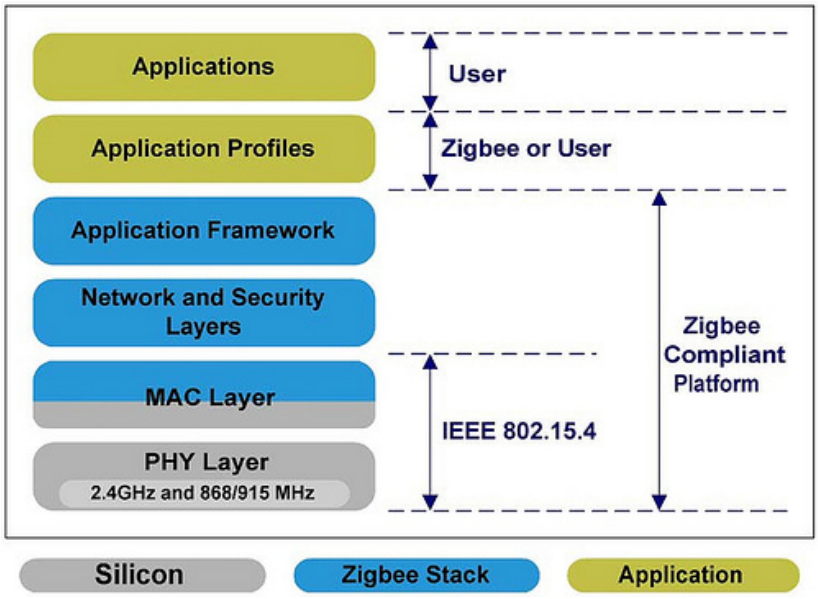



Fig. 3.2 Pila de protocolos de Zigbee

### 3.2.2.Elementos de una red Zigbee

El *Coordinador*. (Siempre tiene que tener uno nuestra red)

Es el nodo de la red que tiene la única función de formar una red. Es el responsable de establecer el canal de comunicaciones y del *PAN ID* (identificador de red) para toda la red. Una vez establecidos estos parámetros, el Coordinador puede formar una red, permitiendo unirse a él a dispositivos *Routers* y *EndPoints*. Una vez formada la red, el Coordinador hace las funciones de Router, esto es, participar en el enrutado de paquetes y ser origen y/o destinatario de información.

Los *Routers*.

Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para enrutar un paquete de información. Lógicamente un Router debe unirse a una red Zigbee antes de poder actuar como Router retransmitiendo paquetes de otros routers o de *EndDevices*.

*EndDevices*.

Los dispositivos finales no tienen capacidad de enrutar paquetes. Deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un Router, es decir, no puede enviar información directamente a otro EndDevice. Normalmente estos equipos van alimentados a baterías. El consumo es menor al no tener que realizar funciones de enrutamiento. Cómo coordinador de nuestra red usaremos el Meshlium y la Waspmote como EndDevices.

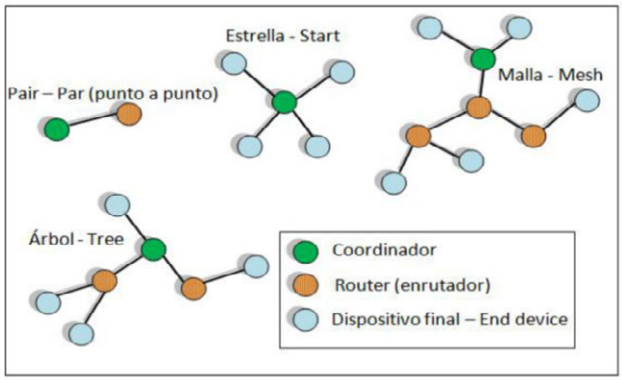



**Fig. 3.3 Topologías de red**

### 3.2.3.Direccionamiento redes Zigbee

Es una dirección de 64 bits estáticos que se asigna a un modulo Xbee y se garantiza que sea único. La Dirección Extendida a veces no se utiliza en Zigbee (para ahorrar memoria). Esta dirección de 64 bits se utiliza para añadir robustez al estándar Zigbee. (4)

Dirección de red.

La dirección de red es una dirección de 16 bits que se asigna por el Coordinador cuando un nodo se une a una red. La dirección de 16 bits debe ser única para una determinada red para que el nodo este debidamente identificado. El problema es que si un nodo fuera a caer fuera de la red (se apagará, ponerse fuera de cobertura, etc.) podría perder su dirección de 16 bits. Es por eso que mantenemos tablas de la red y las direcciones extendidas.

Cuando se envía un mensaje de la dirección extendida es absolutamente necesaria, mientras que la dirección de red es una conveniencia. Al no conocer la dirección de red hará de módem para realizar un descubrimiento de direcciones de red, lo que puede añadir el tiempo de latencia.

*Personal Area Networkd.* Se entiende por PAN una red de comunicaciones que incluye un Coordinador de red y uno o más routers o dispositivos finales (EndPoints).

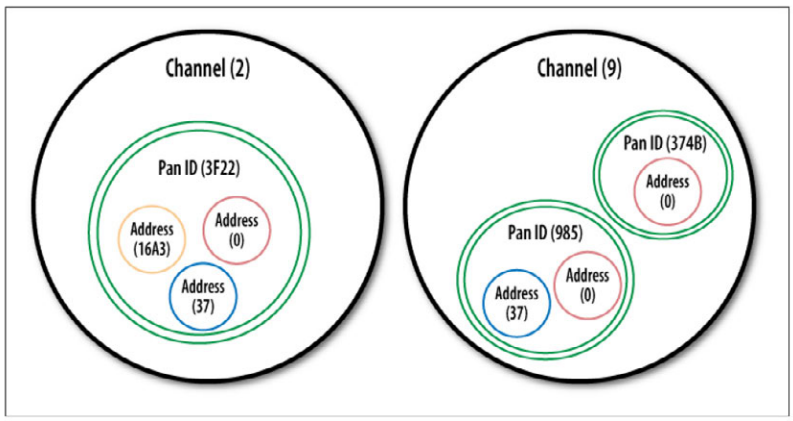



Fig. 3.4 Ejemplo de direccionamiento Zigbee

### 3.2.4.Nivel físico

Xbee, es así como se conoce a los módulos a nivel físico, son fabricados por Digi, una de las empresas que formaron parte de la Zigbee Alliance, la decisión de que modulo utilicemos condicionara mucho la red que deseemos realizar, por eso comenzare a explicar bien esto. Pero antes unos conceptos de radio.

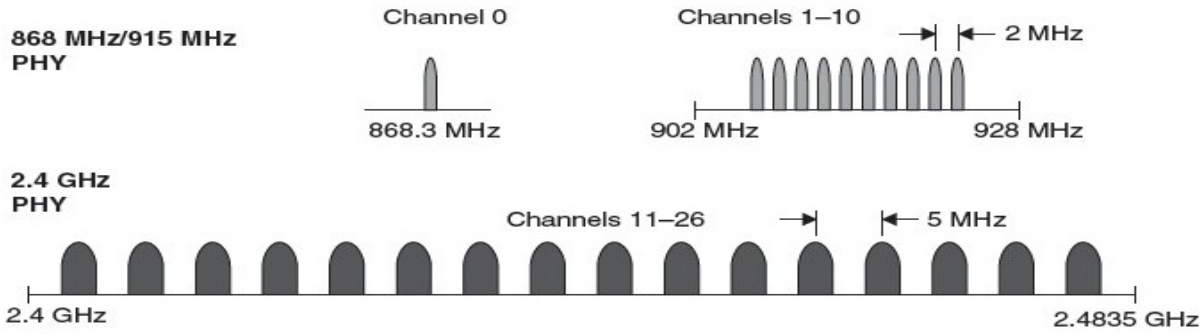



Fig. 3.5 Canales de radio para Zigbee

Zigbee opera en la banda de los 2.4GHz, ocupando en la misma 16 canales de radio, del 11 al 26 (Fig. 3.5), los cuales tienen una anchura de 5 Hz cada uno. Puede calcularse la frecuencia de centro para cada canal con la siguiente fórmula: f = (2350 + 5·ch), donde ch es el canal y, como se ha indicado, puede tomar valores de 11 a 26. Obviamente, un dispositivo sintonizado en un canal de radio no puede transmitir mensajes a dispositivos que se encuentran en otro canal ni tampoco escuchar mensajes de dichos dispositivos.

### 3.2.5.Tipos de Módulos Xbee

Existe una gran variedad de módulos, pero se separan en dos clases, que NO son compatibles:

*Series 1*(también llamados Xbee 802.15.4) - Son la sería más fácil para trabajar, no necesitan ser configurados, pero incluso así se pueden obtener beneficios. Para comunicaciones Punto-a-Punto

*Regular vs Pro* - Hay pocas diferencias entre un Xbee regular y un Xbee PRO. Los XBee PRO son un poco más largos, consumen más potencia y cuestan un poco más de dinero. Eso es casi todo. El mayor consumo de potencia quiere decir también mayor alcance (1,6 Km en vez de 91,5 m). Los dos modelos se pueden mezclar dentro de la misma red.

*Serie 2* (también llamados *ZB*)-Estos son los divertidos. Los módulos Serie 2 deben ser configurados antes de ser usados. Pueden funcionar en modo transparente o por medio de comandos API .Permiten topologías mesh.

*Regular vs Pro*.-Ocurre lo mismo, mas alcance.

La diferencia más importante entre las series es saber que topología queremos implementar.

Tipos de antenas:

Chip Antena – Básicamente es un pequeño chip que actúa como antena. Rápido, sencillo y barato.

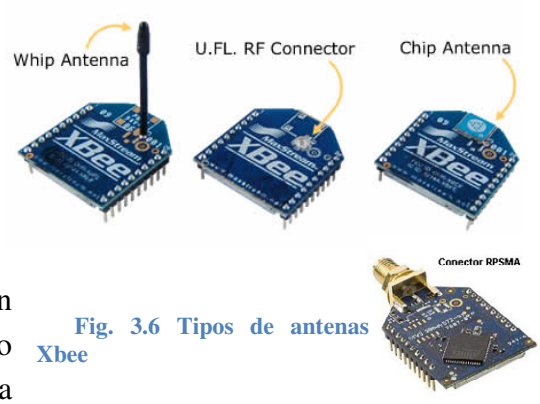



**Fig. 3.6 Tipos de antenas Xbee**

*Wire Antena* (Whip Antenna) – Es un pequeño cable que sobresale, se puede orientar o en caso de encapsular el dispositivo que pueda permanecer fuera.

*u.FL Antena* – Un conector pequeño para conectar tu propia antena. Esto es perfecto si tienes tu equipo en una caja y deseas la antena afuera de ésta.

*RPSMA Antena* – Un conector más grande para conectar tu propia antena, con mas ganancia etc.

Una pregunta llegados a este punto, es si existen distintos módulos Xbee diferentes, ya sean coordinadores, routers o dispositivos finales. La respuesta es NO, somos nosotros los que configuraremos el comportamiento del modulo.

### 3.2.6.Configuración del modulo mediante X-CTU

A continuación se detallaran los pasos a seguir para la configuración del modulo Zigbee. Obtención de programa X- CTU:

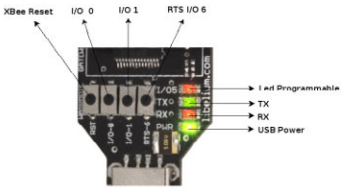



Fig. 3.8 Botones del adaptador USB para XBEE

Fig. 3.7 Modulo Xbee y adaptador unidos

http://www.digi.com/support/kbase/kbaseresultdetl?id=2125

X- CTU fue desarrollado por Digi y que sólo está disponible para Windows. (5)

Instalación de X-CTU: Una vez se ha descargado, el siguiente paso es la instalación del programa. Cuando el programa solicita la actualización de Digi, debemos responder "sí" con el fin de descargar todas las versiones de firmware para todos los módulos Xbee. Cuando X- CTU se ha instalado correctamente, el adaptador con el modulo (Fig. 3.7) puede ser conectado a la computadora (imagen de la derecha). Se reconocerá como un "USB Serial Port". Tenemos que saber el número COM de este dispositivo con el fin de especificar en el X-CTU (en nuestra prueba, COM10 es el valor dado por Windows).Lanzamos X-CTU y el programa comenzará. Aparecerá una ventana como la siguiente (Fig. 3.10), que muestra las diferentes funciones y los diferentes COM detectados.Si el puerto COM para la Waspmote no aparece automáticamente, hay que agregarlo manualmente. Para agregar un puerto COM tenemos que ir a la pestaña *PC Settings*, y cambiar a la pestaña *User Com Ports*. Hay una pequeña caja para agregar Puertos Com *Port Number*, por lo que se sumará el puerto como se muestra a continuación (Fig. 3.9).

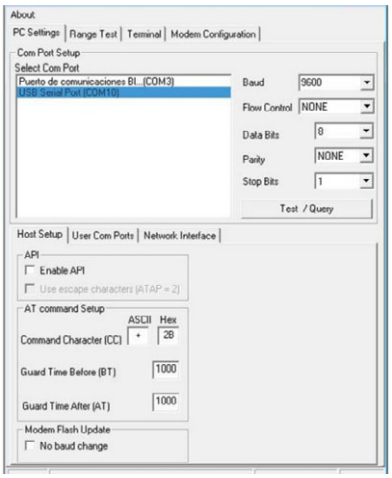



Fig. 3.10 Vista inicial de X-CTU

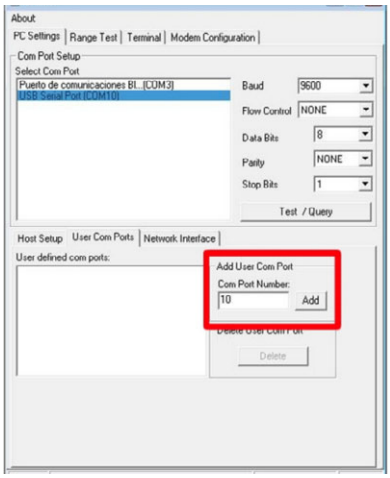



Fig. 3.9 Selección de puerto

Detalles para operaciónes con X- CTU :

Para poder comunicarse con el modulo es necesario configurar la velocidad, en nuestro caso 115200Baudrates. A continuación, conecte el adaptador con el modulo a un puerto USB en su PC, seleccione el puerto de comunicaciones apropiado y configurarlo como se muestra en (Fig. 3.12).Realizar la prueba, pulsamos el botón *Test/Query* y comprobar la aplicación envía un mensaje similar a éste, como podemos observar la aplicación nos informa de que existe comunicación con el modulo, el tipo de modulo, la versión de firmware y el número de serie.

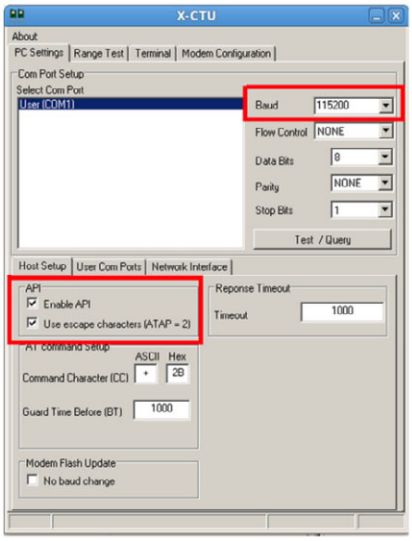



Fig. 3.12 Opciones de comunicación

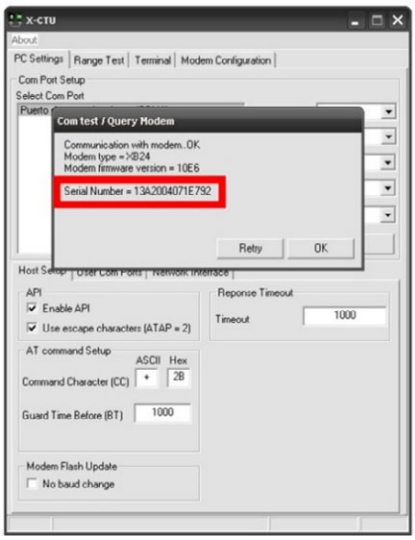



Fig. 3.11 Confirmación de conexión

Es posible que el X- CTU le pide que haga un reset del Xbee en este paso (o en los pasos siguientes). Para ello, sólo tiene que pulsar el botón "RST " (Fig. 3.8) en el adaptador del modulo durante 1 o 2 segundos. A continuación, vaya a la pestaña *Modem*

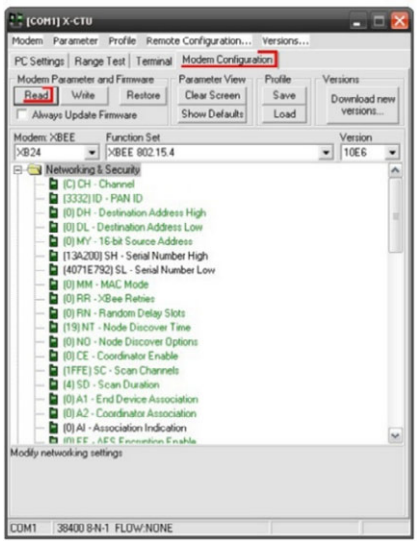



Fig. 3.14 Leer los valores pre-configurados

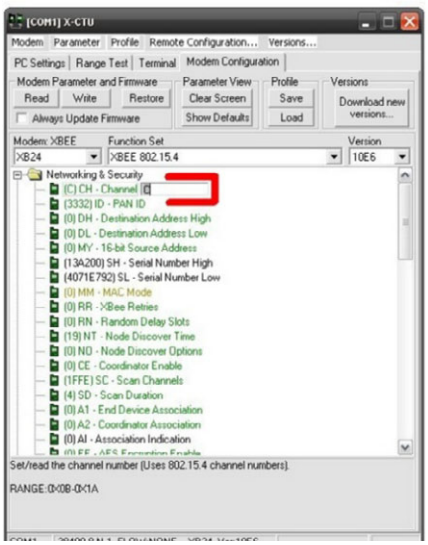



Fig. 3.13 Modulo Xbee, canal de comunicación

*Configuration* y seleccione la opción *Read* (Fig. 3.14). Se cargara la configuración pre-grabada anteriormente en el modulo y podremos seleccionar el parámetro para su configuración. Como vemos existe una gran variedad, en este tutorial explicaremos los utilizados en nuestro caso.

Comprobar el primer parámetro, *CH*: El canal de radio utilizado por nuestra mota será dado por el Meshlium, que será nuestro coordinador en nuestra PAN así que en la configuración inicial del modulo el parámetro *CH* será 0 (Fig. 3.13). El *PAN ID* por defecto para los dispositivos  Libelium es 3332 (Fig. 3.16).

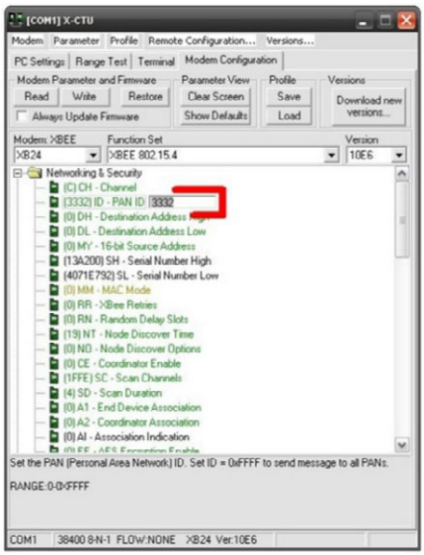



**Fig. 3.16 Configuración direccionamiento Xbee**

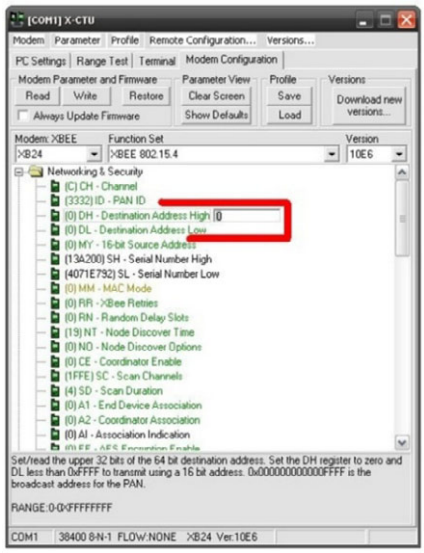



**Fig. 3.15 Configuración dirección destino**

La dirección de destino se introduce en dos partes, *Destination Address High (DH)* y *Destination Address Low (DL)* (Fig. 3.15).Solo será necesaria la parte baja de la dirección.

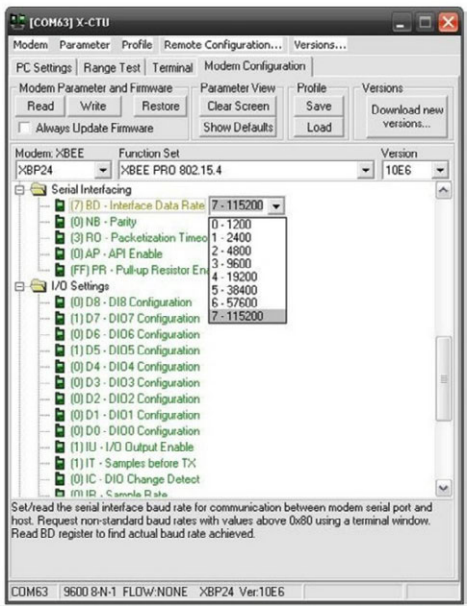



**Fig. 3.17 Velocidad del puerto serial**

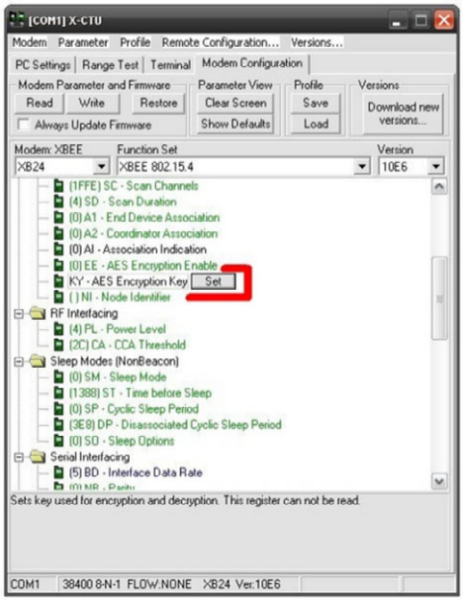



**Fig. 3.18 Clave de encriptación**

En la parte de configuración del Meshlium se explicara cómo obtener su dirección.

El parámetro KY (si es necesario). Se debe establecer como clave hexadecimal, en nuestro caso deshabilitamos esta opción, ya que el encriptar la trama de datos nos quita capacidad de transmisión (Fig. 3.18).

La *Interface Data Rate* como hemos visto al principio es la velocidad de comunicación, esta es configurada a 115200Baud, ya que es la utilizada por nuestra Waspmote para comunicarse con el modulo. En el parámetro *API Enabled* hay que seleccionar la opción *API ENABLED W/PPP* (Fig. 3.19).

Por el ultimo el parámetro *JV*, este es configurado como 0, eso significa que nuestro modulo no tiene asignado un coordinador. Esto se hace así ya que si existe una red Zigbee desplegada en un determinado lugar y se añaden nuevos dispositivos, solo es necesario configurar el modulo de esta manera y colocar la dirección del coordinador como hemos visto anteriormente.

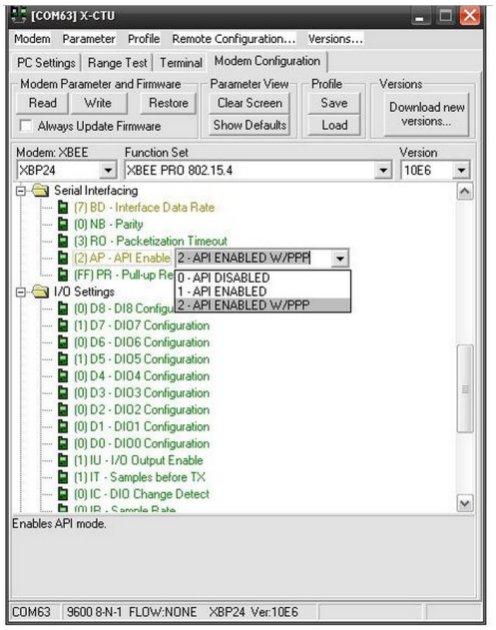



**Fig. 3.19 Habilitar el API**

Pulsamos *Write* y ya podemos cerrar la aplicación y desconectar el USB de forma segura. El modulo está listo para conectarlo a la Waspmote.

El proceso de conexión es el siguiente, el modulo una vez encendido, en nuestro caso cuando se conecte la Waspmote, recorrerá todos los canales de radio, hasta encontrar en el que se encuentre su coordinador, en este momento configura el parámetro *CH* que antes hemos prefijado como 0 con el canal donde está trasmitiendo su coordinador y fija el *JV=1* que significa que está conectado a una red.

## 3.3.Waspmote

### 3.3.1.¿Qué es?

Waspmote(o mota) es una plataforma de sensores inalámbricos de código abierto, tiene como fin la implementación de redes de sensores facilitando la integración de estos y su comunicación con distintas tecnologías (Fig. 3.20).(1)

Fig. 3.20 Waspmote

Es una evolución de Arduino, su principal ventaja frente a esté, es el bajo consumo, la capacidad de programarlo para que entre en un estado de hibernación o duerma, nos permite optimizar el rendimiento del dispositivo al ser desplegado en escenarios reales.

Waspmote se basa en una arquitectura modular. La idea es integrar sólo las partes necesarias en cada dispositivo para optimizar los costes. Por esta razón todos los módulos interfaces de radio, placas de sensores (boards) tienen asignados unos socket concretos. Las diferentes tecnologías que disponemos para comunicar la mota, 802.15.4, Zigbee, Wifi, Bluetooth, GPRS, 3G, RFID, NFC.

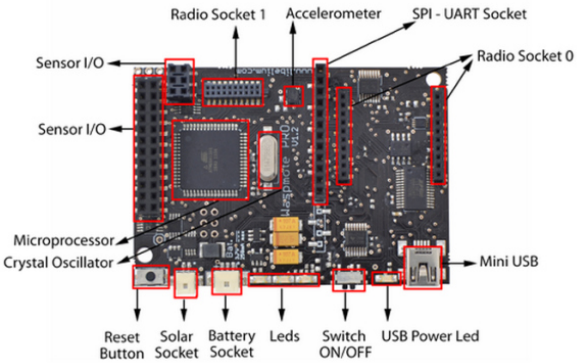



Fig. 3.22 Parte superior de la Waspmote

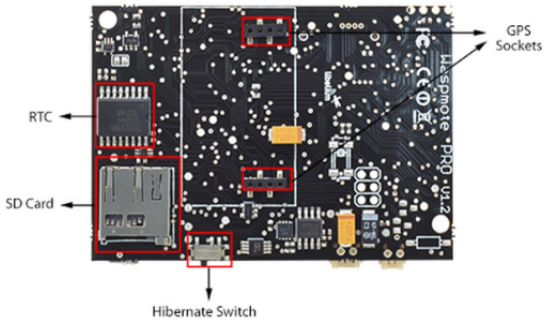



Fig. 3.21 Parte inferior de la Waspmote

Nosotros vamos a utilizar la Waspmote-pro v1.2, sus características son las siguientes.

Micro controlador: ATmega1281:
- Frecuencia: 14MHz
- SRAM: 8KB
- EEPROM: 4KB
- FLASH: 128KB
- SD: 2GB
- Peso: 20gr
- Dimensiones: 73.5 x 51 x 13 mm
- Voltaje de batería: 3.3V-4.2V

Consumo dependiendo de estado:
- On: 15 mA
- Sleep: 55 uA
- Deep Sleep: 55 uA
- Hibernate: 0.06 uA

Hasta ahora hemos definido lo que es el "cerebro" de nuestra mota. Dependiendo de para que aplicación se quiere utilizar existen diferentes placas de sensores (boards). Las boards son placas que proveen al desarrollador de unos socket para sensores predefinidos por el fabricante para unos casos de uso concreto, facilita la instalación de los sensores. Dependiendo de la aplicación a desarrollar encontraremos distintos tipos:

Fig. 3.24 Board para Gases

Fig. 3.23 Board de Eventos

Arriba dos ejemplos, una board orientada a sensores de gases (Fig. 3.24), y la board de eventos (Fig. 3.23) que permite programar cambios de estado. Estas placas se instalan sobre nuestra Waspmote una vez programada y tras instalar el modulo de comunicación Zigbee.

En nuestro caso usaremos para la estación meteorología la board *Agriculture 2.0* (Fig. 3.25) permite monitorizar varios parámetros ambientales que involucran una amplia gama de aplicaciones. Para ello, se ha dotado de sensores para la temperatura del aire, del suelo y la humedad, luminosidad, radiación solar, velocidad y dirección del viento, precipitación, presión atmosférica, humedad de las hojas y la fruta o el tronco de diámetro (dendrómetro). Hasta 15 sensores se pueden conectar al mismo tiempo. Con el objetivo de extender la durabilidad del dispositivo después de la instalación, la placa está dotada de un sistema de interruptores de estado sólido que facilita una regulación precisa de su potencia, lo que prolonga la vida de la batería.

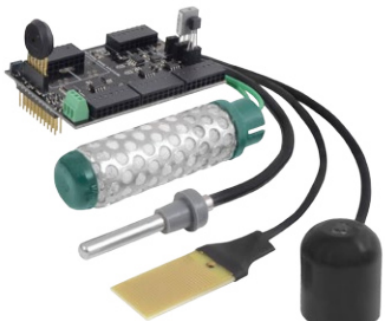


Fig. 3.25 Board de Agricultura

APLICACIONES

- Agricultura de precisión
  La temperatura de la hoja, diámetro del fruto.
- Sistemas de Riego
  La humedad del suelo, humedad de las hojas.
- Invernaderos
  La radiación solar, la humedad, la temperatura.
- Estaciones meteorológicas
  Anemómetro, veleta, pluviómetro.

### 3.3.2.Instalación de sensores

La Board Agriculture 2.0(6) para Waspmote incluye toda la electrónica y sockets necesarios para conectar los sensores más típicos en aplicaciones de agricultura, para nuestro caso utilizaremos aquellos sensores que encajen con el propósito de establecer una estación meteorológica. A continuación detallaremos las características de cada uno y su lugar de instalación en la board.

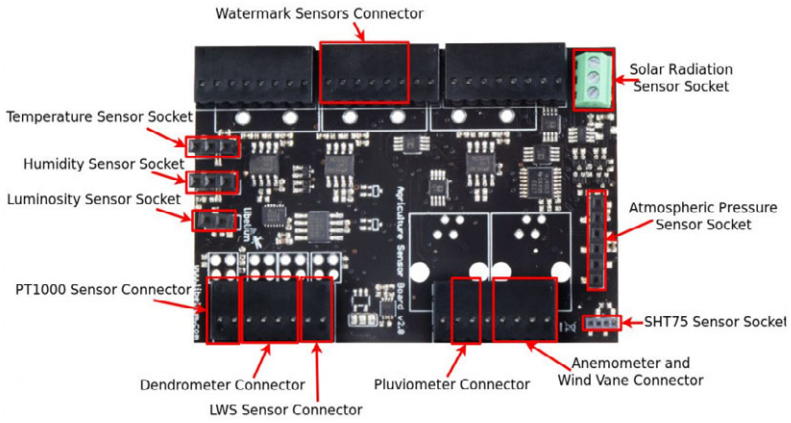



**Fig. 3.26 Board de Agricultura, parte superior.**

Sensor de presión atmosférica (MPX4115A)
Rango de medición: 15 ~ 115kPa
Señal de salida: 0.2 ~ 4.8V (0 ~ 85 º C)
Sensibilidad: 46mV/kPa
Precisión: <± 1,5% V (0 ~ 85 º C)
Consumo típico: 7 mA
Consumo máximo: 10mA
Tensión de alimentación: 4,85 ~ 5.35V
Temperatura de funcionamiento: -40 ~ 125 º C
Temperatura de almacenamiento: -40 ~ 125 º C
Tiempo de respuesta: 20 ms


**Fig. 3.27 Sensor de presión atmosférica**

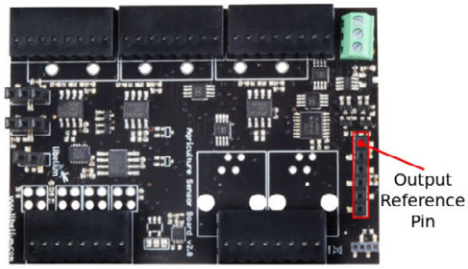



**Fig. 3.28 Posición de instalación del sensor de presión**

Sensor de Humedad (808H5V5)
Rango de medición: 0 ~ 100% RH
Señal de salida: 0.8 ~ 3.9V (25 º C)
Consumo típico: 0.38mA
Consumo máximo: 0,5 mA
Fuente de alimentación: 5VDC ± 5%
Temperatura de funcionamiento: -40 ~ 85 º C
Tiempo de respuesta: <15 segundos
Precisión: <± 4% de humedad relativa (a 25 º C, rango de 30 ~ 80%)


**Fig. 3.29 Sensor de humedad**

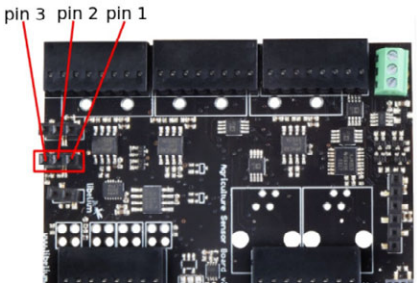



**Fig. 3.30 Posición de instalación del sensor de humedad**

Sensor de Temperatura ((MCP9700A)

Rango de medida: -40 º C ~ 125 º C
Tensión de salida (0 º C): 500 mV
Sensibilidad: 10 mV / º C
Consumo típico: 6µA
Consumo máximo: 12µA
Fuente de alimentación: 2.3 ~ 5.5V
Temperatura de funcionamiento: -40 ~ 125 º C
Temperatura de almacenamiento: -65 ~ 150 º C
Precisión: ± 2 º C (rango de 0 º C ~ 70 º C), ± 4 º C (rango de -40 ~ 125 º C)
Tiempo de respuesta: 1.65 segundos (63% de la respuesta para un rango 30-125 º C)


Fig. 3.31 Sensor de temperatura

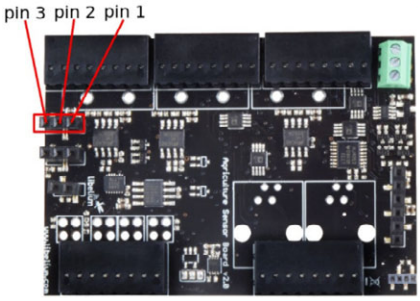



Fig. 3.32 Posición de instalación del sensor de temperatura

Sensor de Luminosidad (LDR)

La resistencia en la oscuridad: 20 mW
Resistencia a la luz (10lux): 5 ~ 20kΩ
Rango espectral: 400 ~ 700nm
Temperatura de funcionamiento: -30 º C ~ +75 º C
Consumo mínimo: 0UA aproximadamente

Fig. 3.33 Sensor de luminosidad

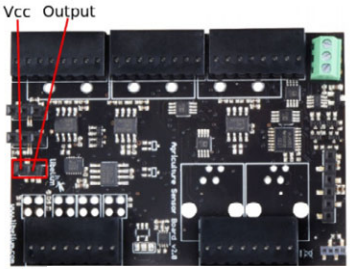



Fig. 3.34 Posición de instalación del sensor de luminosidad

Anemómetro

Sensibilidad: 2,4 kmh / vuelta
Rango de velocidad del viento: 0 ~ 240 kmh
Altura: 7.1 cm
Longitud del brazo: 8,9 cm
Conector: RJ11


Fig. 3.35 Sensor que mide la velocidad del viento

Veleta

Altura: 8.9 cm
Largo: 17,8 cm
Precisión máxima: 22,5 º
Rango de resistencia: 688Ω ~ 120kΩ


Fig. 3.36 Sensor que mide la dirección

Para el anemómetro y la veleta se utilizan los mismos socket, eso es porque la veleta se conecta mediante un conector RJ11 al anemómetro y este último es el que va conectado a

la board, hay que cortar el cable y pelar los distintos cables en su interior para poderlo conectar como aparece en la imagen.

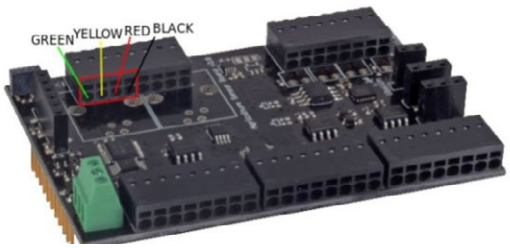



Fig. 3.37 Posición de instalación del anemómetro y veleta

Pluviómetro

Altura: 9.05 cm

Largo: 23 cm

Capacidad de la cuchara: 0,28 mm de lluvia

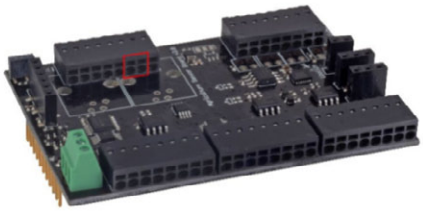


Fig. 3.38 Posición de instalación del anemómetro


Fig. 3.39 Pluviómetro

La instalación de estos últimos tres dispositivos una vez anclados en una base, que da de la siguiente manera (Fig. 3.40).

Fig. 3.40 Estación meteorológica

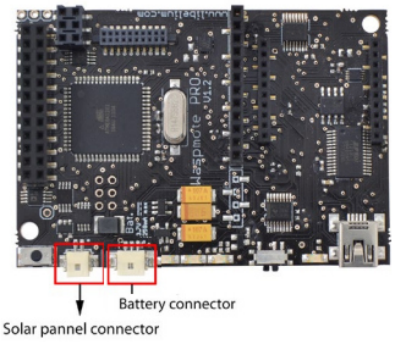
Como se ha querido optar por un sistema autónomo y eficiente, conectaremos una placa solar y una batería a la Waspmote.

Panel solar

7 V - 500 mA

Dimensiones: 234 x 160 x 17 mm

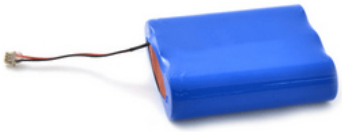
Fig. 3.41 Placa solar




3.42 Conectores para la placa solar y la batería

Batería

Ligera, de alta densidad. Genera 6600mAh, viene con un conector estándar.



Cargar la batería se hace por medio de la Waspmote teniendo está conectada al ordenador mediante el USB.

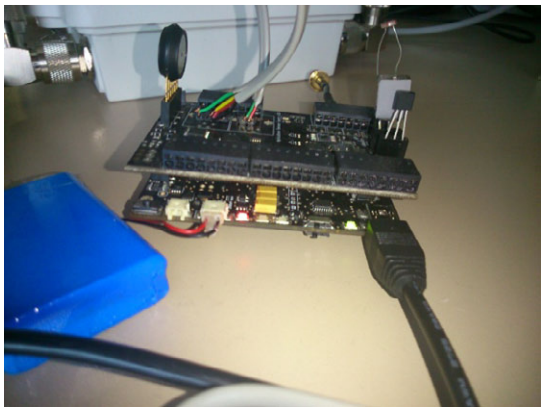


Fig. 3.43 Waspmote con todos los sensores conectados

### 3.3.3.Programación del dispositivo

En este apartado no se quiere hacer un análisis exhaustivo del código, se planteara un diagrama de flujo para hacerse una idea del comportamiento del programa y decisiones que se han tomado en función de problemas y necesidades del escenario.(7)

Pero en primer lugar vamos a ver que necesitamos para programar una Waspmote.

Waspmote IDE

Para poder hacer uso de las distintas funciones que ofrece Waspmote, es indispensable tener instalado previamente el entorno de desarrollo. A través del mismo, generaremos los sketch (así se llaman los ficheros del código) del programa que queramos cargar en nuestro dispositivo.

http://www.libelium.com/development/waspmote/sdk_applications/

Contiene un compilador, gestión de errores, configuración de librerías, monitor serial y pre instala ejemplos de códigos para las diversas funciones de la placa. Una vez compilado el código nos permite exportarlo a la Waspmote, conectándola mediante un cable USB-miniUSB al PC.

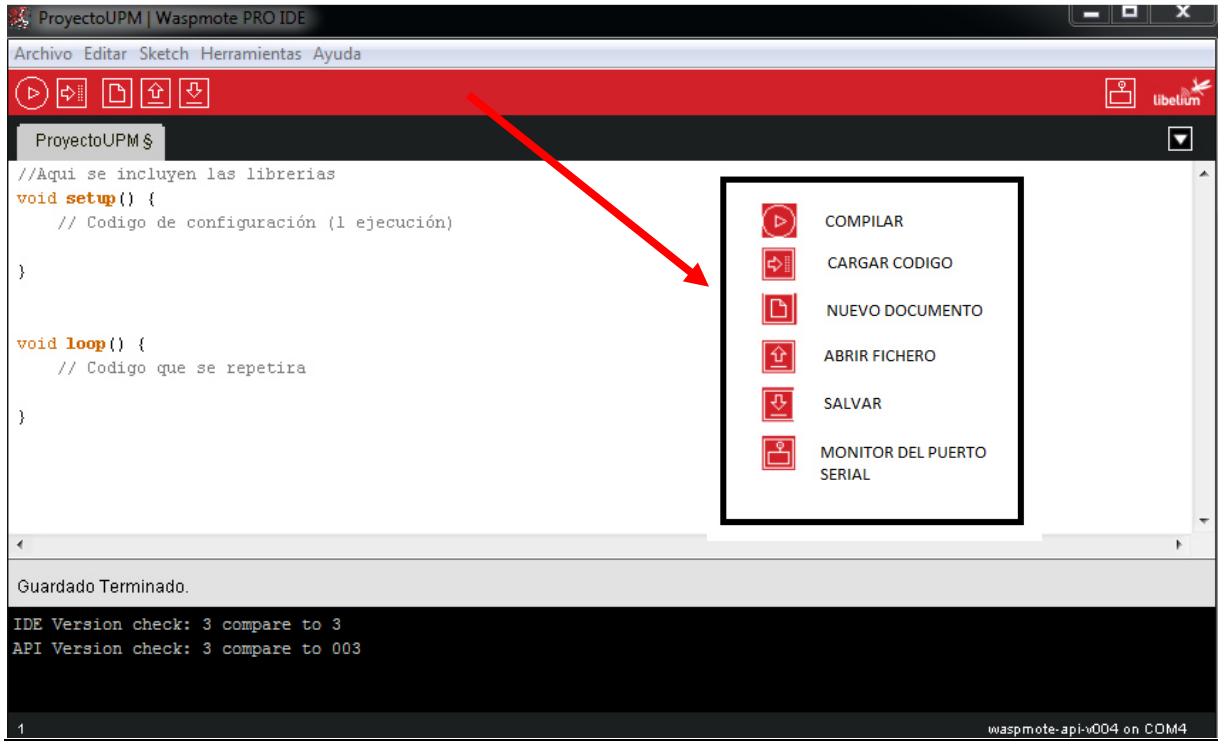



Fig. 3.44 Entorno de desarrollo para Waspmote

Al crear un sketch, aparecerán las funciones '*setup'* y '*loop'*, las cuales debemos incluir siempre, independientemente del programa que estemos desarrollando. Si no es así, al compilar el código nos dará un error.

*Setup:*

Solo se ejecuta una única vez y configura las características generales de la Waspmote, en nuestro caso se encarga de establecer contacto con el Meshlium(coordinador), habilitar el interfaz Xbee y el puerto serial para poder monitorizar y depurar en tiempo real con un PC mediante mensajes intercalados en el código las instrucciones que se van llevando a cabo.

Ejemplos de instrucciones del setup:

*xbeeZB.ON()* //encender el modulo
*USB.ON()* //activar puerto serial

*Loop:*

Esta parte se ejecutara a continuación del setup y durante el resto del tiempo que la mota está encendida. Aquí se encuentra el "cuerpo" del código.

Librerías importantes:

*#include <WaspXBeeZB.h>*

Nos provee de las distintas funciones para una comunicación con el modulo Xbee, activarlo, desactivarlo, estado del buffer y verificar sus estados de transmisión.

*#include <WaspSensorAgr_v20.h>*

Esta Liberia contiene los métodos concretos para nuestra board.

*#include <WaspFrame.h>*

A la hora de gestionar y preparar las tramas de una manera fácil de identificar por el Meshlium, Libelium nos facilita esta clase.

Capturar una medida de un sensor

Hay que entender a la hora de tomar una medida con un sensor, que la ejecución de código es mucho más rápida que cualquier parte física, por ello cada vez que realizamos una acción sobre cualquier elemento del sistema, entiéndase socket para un sensor o interfaz de comunicación hay que respetar ciertos tiempos medidos en nuestro caso en milisegundos para que la parte física pueda ajustarse. Después de ciertas instrucciones es una buena práctica poner una espera:

*delay(1000);//El programa se para en esa línea durante 1seg*

Para que la medida sea correcta en un sensor hay que realizar los siguientes pasos:

1) Habilitar

2) Esperar el tiempo que recomiende el fabricante para cada sensor

3) Leer dato

4) Deshabilitar

A continuación un ejemplo de cómo leer un sensor y almacenar la medida en una trama antes de ser enviada.

Dependiendo del valor que devuelva el sensor tenemos que crear una variable de un determinado tipo, para nuestro ejemplo el sensor de temperatura toma valores decimales, por ello utilizamos un float.

*float TemperatureFloatValue;*

Paso 1, Habilitamos el sensor, en la instrucción se intuye como establecemos el modo de operación de este, diciéndole a las Waspmote que habilitamos el sensor de temperatura

*SensorAgrv20.setSensorMode(SENS_ON, SENS_AGR_TEMPERATURE);*

Paso 2, Tiempo que necesita el sensor para capturar la medida, (medio segundo)

*delay(500);*

Paso 3, Leemos el valor y lo almacenamos en la variable anteriormente creada.

*TemperatureFloatValue = SensorAgrv20.readValue(SENS_AGR_TEMPERATURE);*

Paso 4, Deshabilitamos el sensor

*SensorAgrv20.setSensorMode(SENS_OFF, SENS_AGR_TEMPERATURE);*

Gestión de tramas

Cuando queremos enviar una trama mediante Xbee existen diferentes maneras dependiendo de las decisiones de diseño elegidas. La primera por el modulo y firmware configurado, el nuestro es Xbee-Zigbee, por otro lado vamos a trabajar con datos que no son sensibles y dada la cantidad de sensores necesitaremos al menos dos tramas para mandar una medida de cada sensor, así que prescindimos de encriptación de enlace y de AES a nivel de aplicación. Por tanto nuestra trama es de 74Bytes.

| Module | | | Maximum frame size | |
|---|---|---|---|---|
| | | | No AES encryption | AES encryption |
| XBee – 802.15.4 | Link Encrypted | @16bit Unicast | 98 Bytes | 93 Bytes |
| | | @64bit Unicast | 94 Bytes | 77 Bytes |
| | | Broadcast | 95 Bytes | 77 Bytes |
| | Link Unencrypted | | 100 Bytes | 93 Bytes |
| XBee – 868 | | | 100 Bytes | 93 Bytes |
| XBee – 900 | Link Encrypted | | 80 Bytes | 77 Bytes |
| | Link Unencrypted | | 100 Bytes | 93 Bytes |
| XBee - Digimesh | | | 73 Bytes | 61 Bytes |
| XBee - ZigBee | Link Encrypted | @64bit Unicast | 66 Bytes | 61 Bytes |
| | | Broadcast | 84 Bytes | 77 Bytes |
| | Link Unencrypted | @64bit Unicast | 74 Bytes | 61 Bytes |
| | | Broadcast | 92 Bytes | 77 Bytes |

**Fig. 3.45 Capacidad de una trama Zigbee dependiendo de su topología y encriptación**

No solo tenemos que mandar tramas con las medidas de los sensores, sino que además necesitamos que el Meshlium reconozca los datos y a su correspondiente sensor.

Ayudados por la librería *WaspFrame.h* los datos irán en cada trama de la siguiente manera, primero una etiqueta del sensor definida por Libelium y después el valor tomado de este, así hasta llenar la trama de 74bytes.

| Payload | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SENSOR_STR | Length | "STRING" | SENSOR_BAT | 0x57 | SENSOR_IN_TEMP | 0x00 | 0x00 | 0xDA | 0x41 |
| 0 | 1 | 2-7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**Fig. 3.46 Ejemplo del Datagrama**

Ejemplo de cómo guardar el dato en la trama.

*frame.addSensor (SENSOR_TCA, TemperatureFloatValue);*

Relación de sensores y nombres de variables, el Meshlium reconoce y separa en destino los valores para cada sensor.

| SENSORES | ETIQUETA LIBELIUM |
|---|---|
| Temperatura | SENSOR_TCA |
| Humedad | SENSOR_HUMA |
| Presión Atmosférica | SENSOR_PA |
| Luminosidad | SENSOR_LUM |
| Batería | SENSOR_BAT |

| | |
|---|---|
| Anemómetro | SENS_AGR_ANEMOMETER |
| Pluviómetro | SENS_AGR_PLUVIOMETER |
| Veleta | SENS_AGR_VANE |

**Fig. 3.47 Relación sensores y el nombre con el que son reconocidos por el Meshlium**

Modos de consumo de energía

Waspmote tiene 4 modos de funcionamiento:

*On*: El modo de operación normal. El consumo en este estado es de 15 mA.

*Sleep*: El programa principal está en pausa, el microcontrolador pasa a un estado latente, de la que puede ser despertado por todo

Interrupciones asíncronas y por la interrupción síncrona generada por el Watchdog. El intervalo de duración de este estado es de 32 ms a 8s. El consumo en este estado es 55µA.

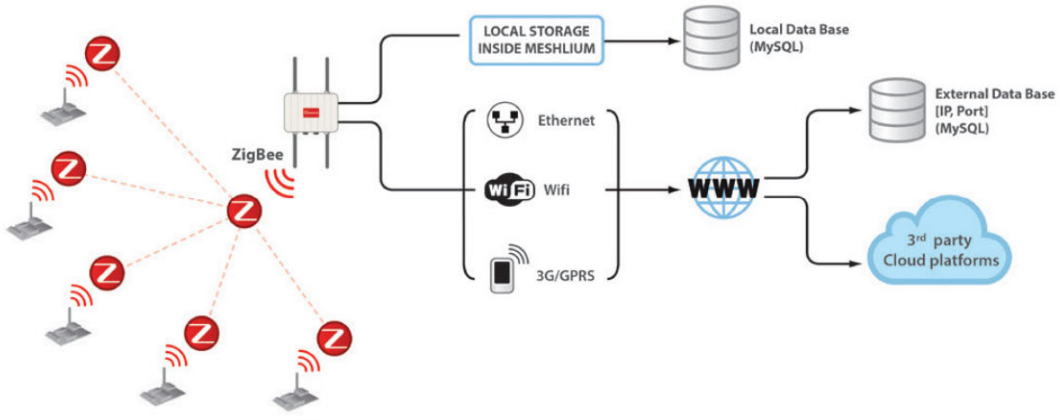
*Deep sleep*: El programa principal se detiene, el microcontrolador pasa a un estado latente de la que puede ser despertada por todas interrupciones asíncronas y por la interrupción síncrona provocada por el RTC. El intervalo de este ciclo puede ser desde segundos a minutos, horas, días . El consumo en este estado es 55µA.

*Hibernate*: El programa principal se detiene, el microcontrolador y todos los módulos de Waspmote son completamente desconectados. La única manera de reactivar el dispositivo es a través de la alarma previamente programada en el RTC (interrupción síncrona). El intervalo de este ciclo puede ser desde segundos a minutos, horas, días. Casi todos los dispositivos están totalmente desconectados de la de la batería: sólo el RTC es alimentado a través de la batería, de la que se consume 0.06µA.

## 3.4.Meshlium

### 3.4.1.¿Qué es?

Meshlium es un router Linux que puede contener 5 interfaces de radio diferentes: Wifi 2,4 GHz, 5 GHz Wifi, 3G/GPRS, Bluetooth y Zigbee. Además de esto también se puede integrar un módulo GPS para aplicaciones móviles en vehículos ya que puede ser alimentado por una batería. Estas características, junto con una IP65 carcasa de aluminio permite Meshlium ser colocado en cualquier lugar al aire libre.(2)




3.48 Esquema global de gestión del Meshlium

Todas las opciones de red pueden ser controladas a partir de dos fuentes diferentes:

Manager System: una interfaz web que viene con Meshlium. Esto le permite controlar todas las interfaces y el sistema opciones en una forma segura, fácil y rápida.

Consola SSH: para usuarios expertos se habilita acceso directo a la consola de Shell.

Toda la información que proviene de todas las interfaces (Zigbee, Bluetooth, 3G/GPRS, Wifi y desde el módulo GPS) se pueden almacenar en el sistema de archivos y la base de datos local, trae instaladas dos tipos de bases de datos, MySQL y PostgreSQL. Otra opción también podría ser configurar el dispositivo para que reenvié los datos a otro servidor.

Dada su capacidad de procesamiento por ser un ordenador y la variedad de tecnologías de comunicación le hacen un elemento a tener encuentra a la hora de las redes de sensores, en nuestro caso actuara de coordinador de la red Zigbee, almacenado y gestionando las tramas enviadas por nuestra Waspmote, pero este router está preparado para poder administrar un número considerable de dispositivos.

Especificaciones técnicas generales

3.49 Meshlium

Procesador a 500 MHz (x86)
256 MB de memoria RAM (DDR)
La memoria de disco 16 GB
Fuente de alimentación PoE (Power over Ethernet)
Dimensiones 210x175x50mm
Peso: 1,2 Kg
La protección exterior IP65
Rango de temperatura de -20 º C / 50 º C
Tipos de fuente de alimentación AC-220V,Batería - panel solar (DC-12V)
Sistema Linux, Debían. Comunicación OLSR Mesh protocol.
Software de gestión Meshlium manager system (código abierto)
Seguridad Autenticación WEP, WPA-PSK, HTTPS y Acceso SSH.

Especificaciones de los interfaces

Cuando se adquiere un dispositivo de este tipo, se puede elegir entre una variedad de antenas, dependiendo de las tecnologías a abarcar y sus frecuencias.

WIFI RADIO

Chipset Atheros AR5213A - IEEE 802.11b / g
Potencia de transmisión 100mW - 20dBm
Distancia 500m (en función de la línea de visión)
Antena
Tipo omnidireccional
Ganancia 5dBi
Dimensiones 224x22mm

ZIGBEE

Modelo Xbee - PRO - Zigbee
Frecuencia 2,4 GHz
Potencia de transmisión 50mW
Sensibilidad del receptor 102dBm
Antena 5dBi dipolo
Distancia 7 kilometros

MODULO 3G/GPRS

Protocolos 3G, WCDMA, HSPA, UMTS, GPRS, GSM
Bandas de frecuencia 850MHz/900MHz/1900MHz/2100MHz
Potencia de salida
UMTS 850/900/1900/2100: 0,25 W
GSM850/GSM900: 2W
DCS1800/PCS1900: 1W
Rx 7.2Mb/s
Antena 3dBi

MODULO GPS

Modos de GPS asistido (A-GPS),
Antena 26dBi (+-4.5dBi) - Cable de 3m. Magnético

### 3.4.2.Instalación

Existen diversas formas de instalación de Meshlium, dependiendo del uso que se le vaya a dar, en nuestro caso al querer comunicarnos para poder configurar y testear diversas configuraciones elegimos alimentar el dispositivo mediante un cable Ethernet, así además podremos comunicarnos y poder administrar el Meshlium desde un PC. Señalar que una vez establecida una primera conexión se puede habilitar el interfaz Wifi y prescindir del cable Ethernet.

Fig. 3.50 Esquema de montaje para su configuración.

El sistema tiene que estar conectado de la siguiente manera y se realizara en un orden con el fin de dañar el Meshlium.

Conectamos el cable de red cruzado que se incluye en caja, (tiene una etiqueta de identificación) a la entrada POE marcada en el adaptador.

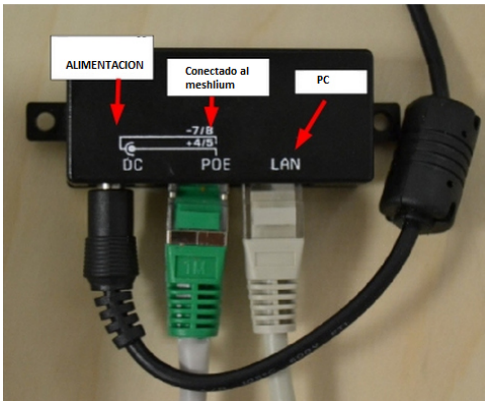



Fig. 3.52 Cableado

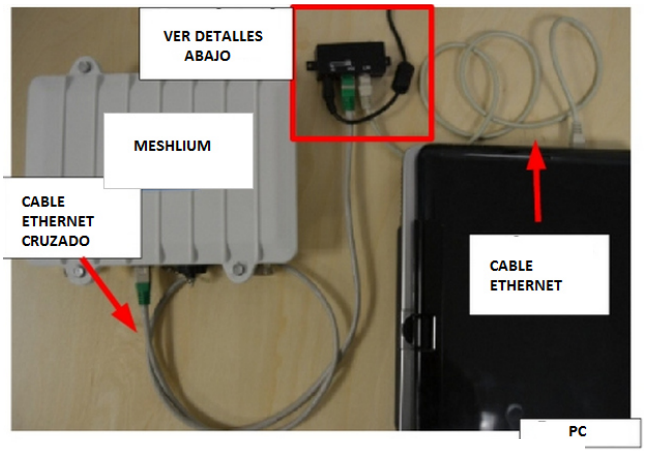



Fig. 3.51 Disposición de los elementos

El cable estándar de Ethernet se conecta en el adaptador donde aparece señalado LAN y el otro extremo al PC. Por último alimentamos el adaptado conectando el cargador en DC y ya podemos enchufarlo a red eléctrica.

Una vez alimentado tras unos segundos se escuchara un “pitido”, eso quiere decir que el equipo ha arrancado, es importante esperar unos 2-3 minutos antes de entablar una comunicación con el dispositivo.

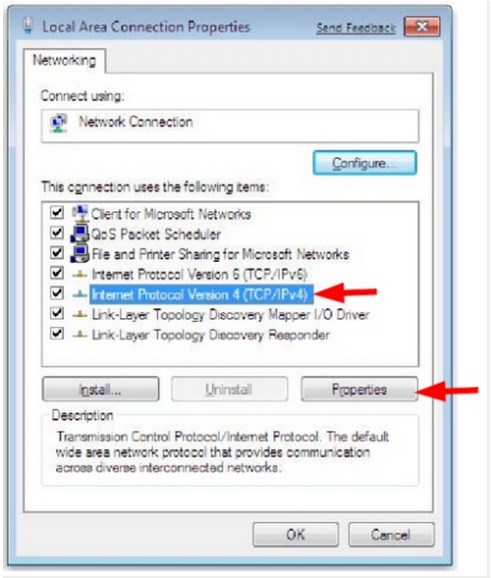
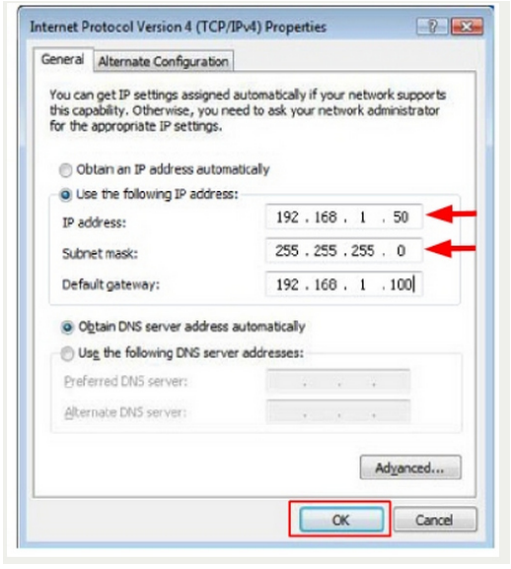



Fig. 3.54 Ventana de propiedades del área local




3.53 Configuración con los valores de red del Meshlium

Para poder comunicarnos a través del ordenador es necesario editar la configuración de Ethernet. Para ello nos dirigimos al Panel de control → Conexiones de red e Internet → Cambiar configuración del adaptador. Ahora haga clic derecho en su conexión Ethernet, haga clic en la opción Propiedades. Elegimos la configuración para protocolo de internet versión 4 y acto seguido a propiedades (Fig. 3.54). Configurar con los valores que aparece en la imagen (3.53).

Relación de antenas y su conexión en el Meshlium

Este paso se puede llevar a cabo con el Meshlium encendido, todas las antenas como se ve en las imágenes detallan las frecuencias y modelo que son. Nosotros incluiremos unos codos adaptadores que nos permiten mantener las antenas verticales ya que lo vamos a colocar en el laboratorio sobre una superficie y no en la pared.


Fig. 3.55 Conector y antena Wifi


Fig. 3.56 Conector y antena GPRS


Fig. 3.57 Conector y antena Zigbee

Fig. 3.58 Conector y GPS

### 3.4.3.Configuración del sistema

Tras la instalación del entorno y configurada nuestra tarjeta de red estamos listo para acceder con nuestro navegador a esta dirección:

http://192.168.1.100/ManagerSystem

Introducimos usuario y la contraseña por defecto en la pantalla de inicio (Fig. 3.59).

User: root
Password: libelium


Fig. 3.59 Pantalla de inicio

La Pantalla principal del menú para configurar el sistema una vez nos hemos autenticado correctamente (Fig. 3.60), en la parte superior tenemos las diferentes secciones, interfaces, sensor networks, tools, system, update manager, help (Fig. 3.61).

Arriba a la derecha encontramos el botón de reinicio y apagado, al igual que el arranque es un momento crítico para la estabilidad del equipo, así que es necesario una vez pulsado el botón de apagado, esperar a escuchar unos pitidos tras 2-3min que aseguran que podemos retirar la alimentación sin alterar el sistema de arranque.

En el reinicio, primero un pitido de apagado 2-3min de espera, y a continuación otro de encendido y 2-3min de espera para poder acceder al dispositivo.


Fig. 3.60 Menú principal

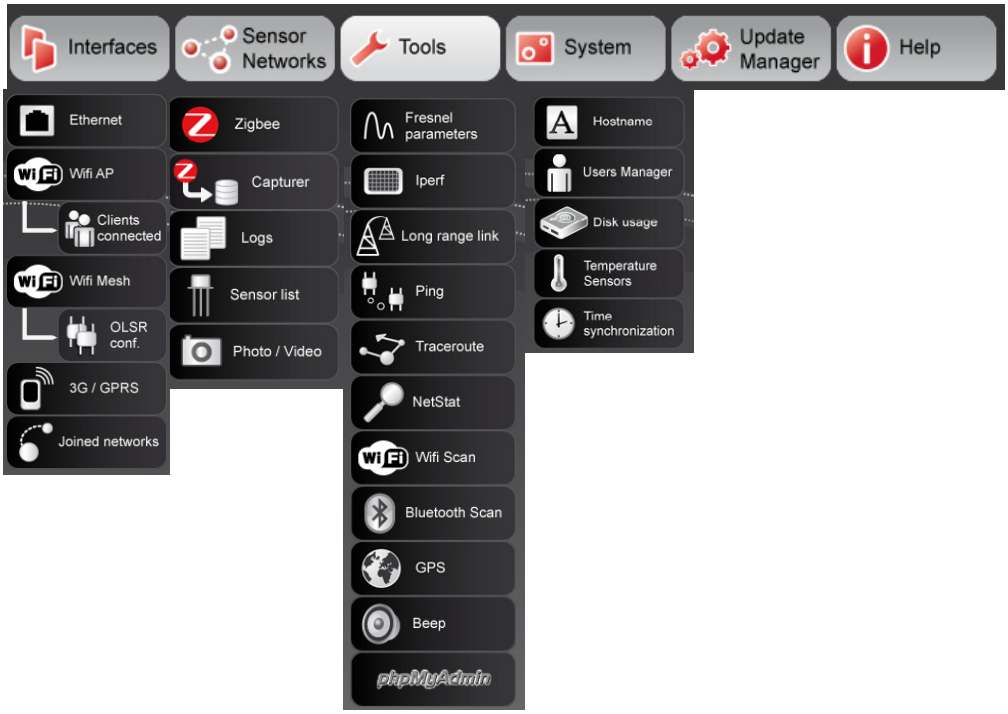



Fig. 3.61 Menú principal desplegado

Nos centraremos en la secciones de *Interfaces* y *Sensor Network* en su totalidad. En la sección de *System* solo se han reconfigurado parámetros básicos de administración. En *Tools* solo es necesario configurar el GPS y *phpMyAdmin* para la base de datos.

### 3.4.4.Interfaz Wifi

Tener configurado la red Wifi, nos permite prescindir del cable de Ethernet al PC en el caso de estar utilizando el notebook y poder acceder al Meshlium para su mantenimiento y configuración a distancia. En la imagen se pude ver la configuración óptima.

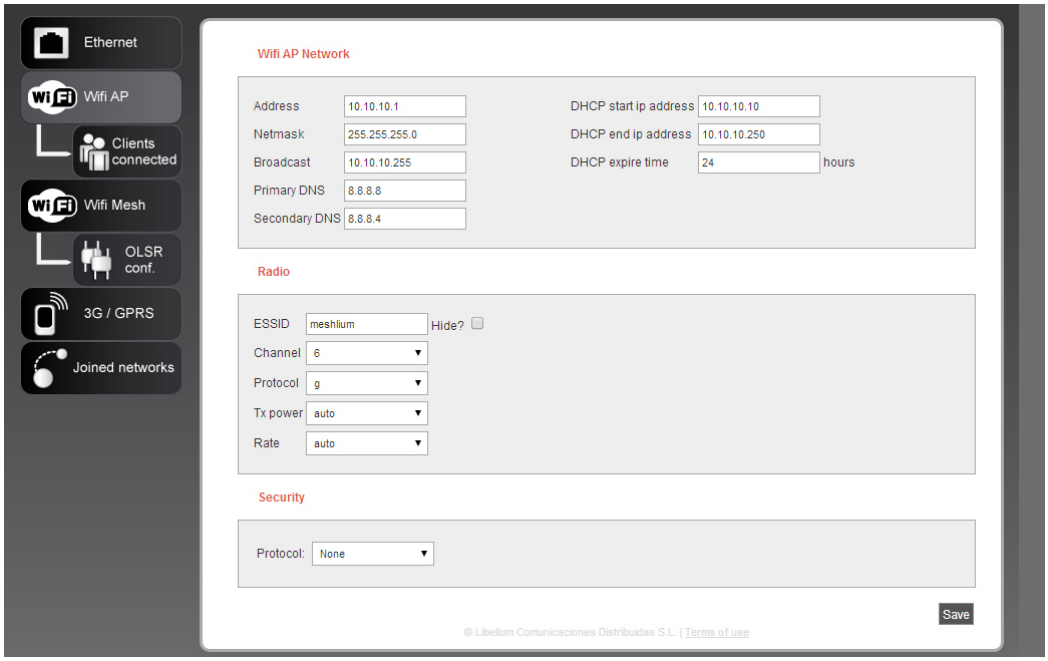
I 


Fig. 3.62 Menú de configuración para Wifi

Para acceder mediante este interfaz, bastar con activar una tarjeta inalámbrica en un PC, buscar la red con ESSID Meshlium, en nuestro caso no requiere autenticación de cifrado.

Accedemos mediante el navegador:

http://10.10.10.1/ManagerSystem

El usuario y la contraseña por defecto los mismos que por el interfaz de Ethernet:

User: root
Password: libelium

Una vez conectados por Wifi, desconectamos el cable Ethernet. Se aconseja por seguridad modificar la configuración Wifi inicial y encriptarla además de cambiar las contraseñas por defecto.

### 3.4.5.Interfaz GPRS

La instalación de una tarjeta SIM solo puede realizarse por Libelium si no quiere perderse la garantía del equipo. Hay que configurar con los valores propios del proveedor de servicio.

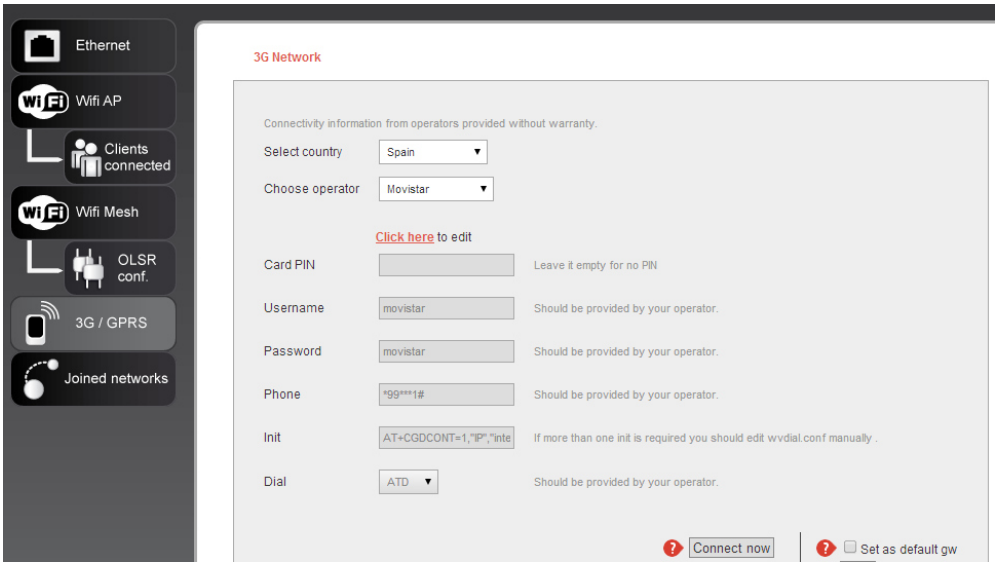
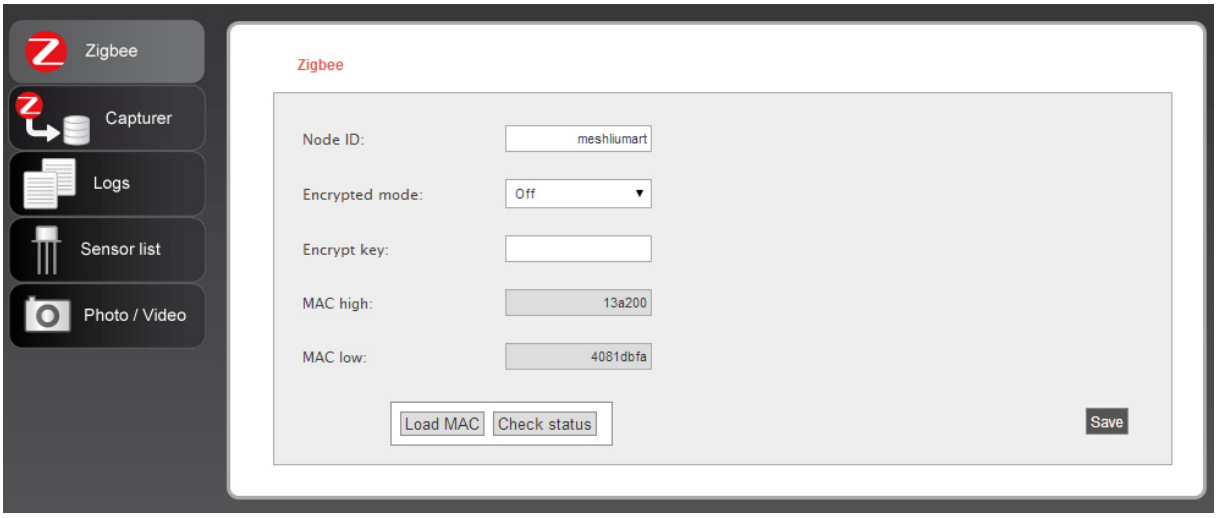



Fig. 3.63 Configuración de interfaz de GPRS

### 3.4.6.Interfaz Zigbee

Aquí podemos ver la dirección del Meshlium, anteriormente es la que hemos utilizado para la configuración del modulo Xbee en la Waspmote como dirección destino, es decir como dirección a conectarse, esta dirección al igual que las direcciones MAC de Ethernet es única para cada dispositivo.




3.64 Configuración Zigbee

### 3.4.7.Base de datos

El almacenamiento de información del Meshlium cuenta con dos sistemas de bases de datos diferentes.

MySQL (por defecto)

Postgres

Toda la información que proviene de todas las interfaces (Zigbee, Bluetooth, 3G/GPRS, Wifi y desde el módulo GPS) se pueden almacenar en el sistema de archivos local y la base de datos local configurando "Storage Options" o exportar datos a otra máquina conectada a la Internet.

### 3.4.8.Arranque del Sistema, establecer comunicación Meshlium-Waspmote

Se van a detallar los pasos para que la conexión entre la Waspmote y Meshlium se lleve a cabo con éxito, además el Meshlium gestionara las tramas, separando la información para cada sensor, esto se denomina *parseador de tramas*, en el apartado (Waspmote/Programación del dispositivo) como se explicaba en la parte de programación, los datos se envían de una determinada manera al Meshlium, para que este los reconozca y pueda gestionarlos.

Desde el *ManagerSystem*, accedemos al interfaz de Zigbee, hacemos click en *Chech Status* (Fig. 3.65), si todo está configurado correctamente nos aparecerá esto:

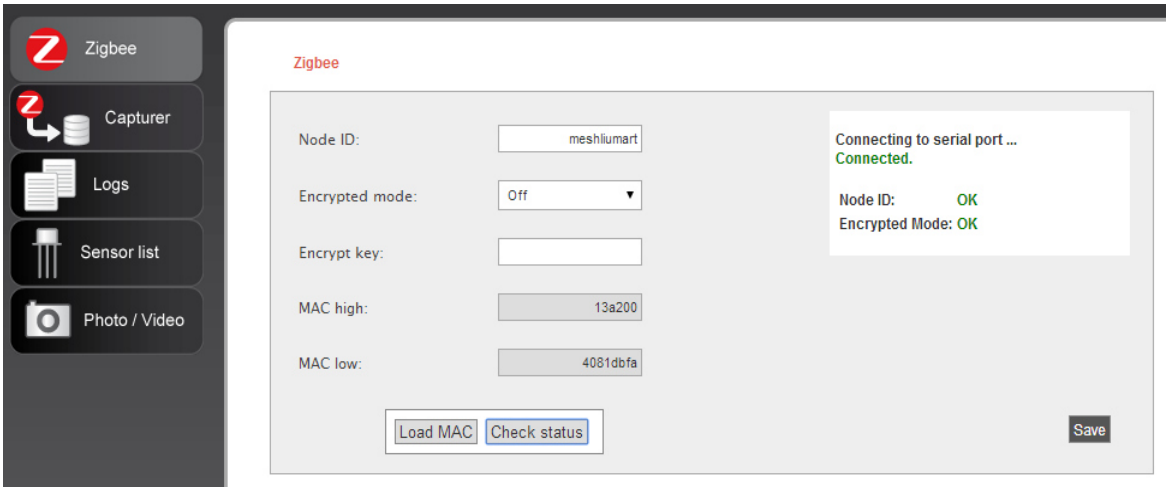



Fig. 3.65 Resultado tras pulsar el chequeo del estado

Dentro de la sección de Zigbee en el apartado *Capturer* nos aparece arriba a la derecha "*Sensor Parser Not Available*" (Fig. 3.66), esto significa que aunque el Meshlium reciba y almacene nuestras tramas, estas no serán gestionadas por sensores individuales. Es decir, no está pendiente del contenido de las tramas y diferenciar los datos de cada sensor. Para activar el parseador es necesario activar unos script.

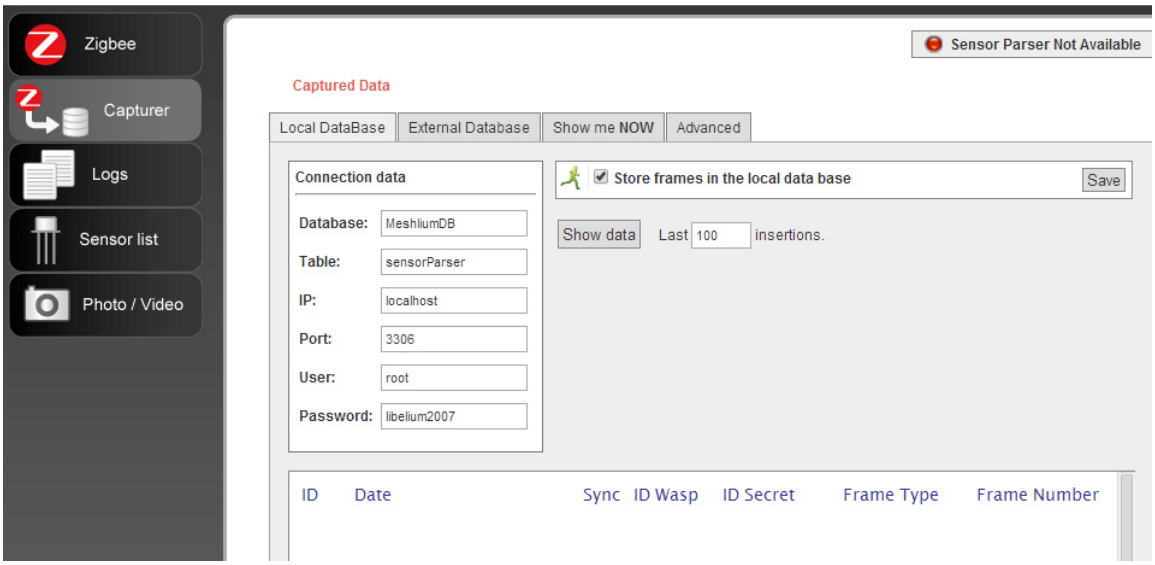



Fig. 3.66 Pantalla de captura de datos

### 3.4.9.Habilitar el parseador de datos

Para ello necesitamos descargarnos si estamos utilizando un sistema Windows un cliente ssh, en la mayoría de las distribuciones de Linux viene instalado por defecto en el terminal.

Descargamos y configuramos (Fig. 3.67) el *PuTTy*, cliente ssh para Windows de la siguiente manera una vez inicializado.

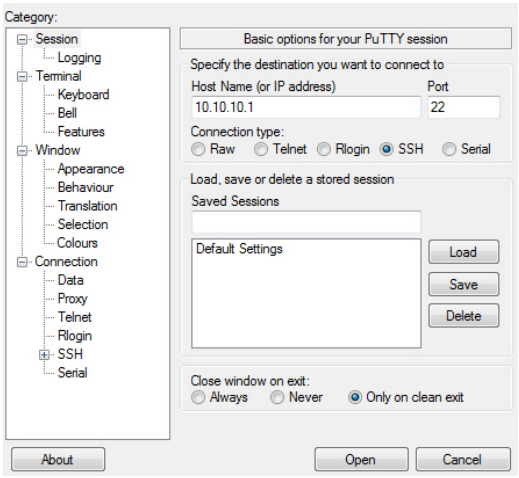



Fig. 3.67 Ventana de configuración del Putty

Se nos abrirá una consola, lo primero que nos preguntara será, el usuario y contraseña:

Login as: root

Password: libelium (esta parte no es visible por el usuario, escribimos y *enter*)

Tras esto el sistema nos da la bienvenida, hemos accedido al sistema de un router Linux (Fig. 3.68), es recomendable saber lo que se hace ya que ahora cualquier cambio no es gestionado por el *Manager System* y podemos dañar el software pre instalado.

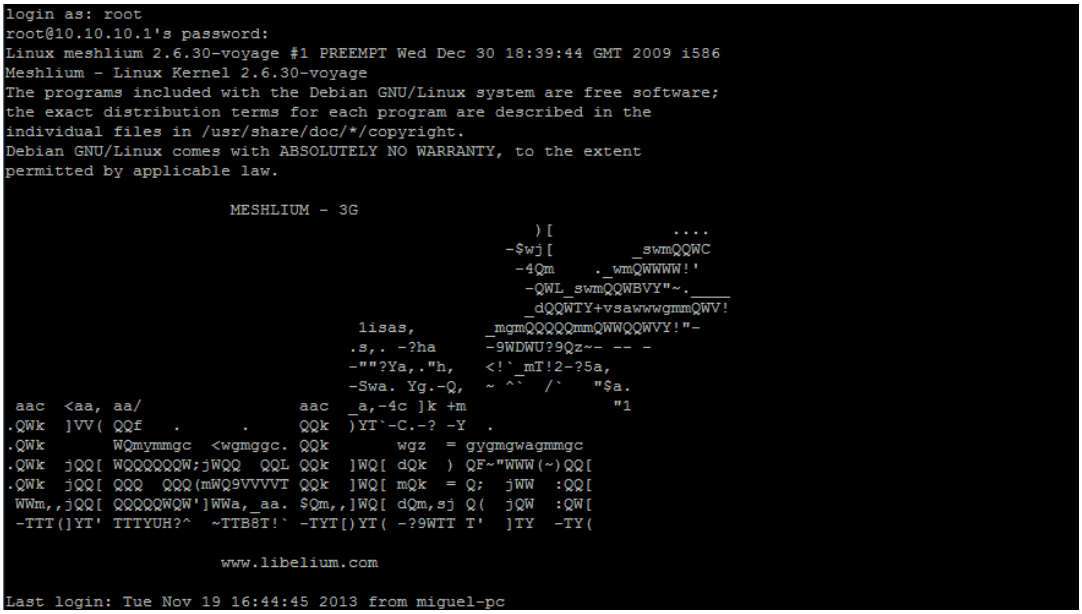



Fig. 3.68 Pantalla de inicio de sesión por ssh en el Meshlium

Ya están incluidos dos script dentro del sistema, solo es necesario ejecutarles cada vez que se reinicia el Meshlium. Escribiendo las siguientes instrucciones por separado y pulsando la tecla *Enter*.

*ZigbeeScanD.sh start*
*sensorParser.sh start*

Al ejecutar la segunda instrucción, el Meshlium contesta y vemos en pantalla que se ha iniciado el parseador. Podemos cerrar el terminal directamente. Hemos dejado corriendo el parseador (Fig. 3.69).

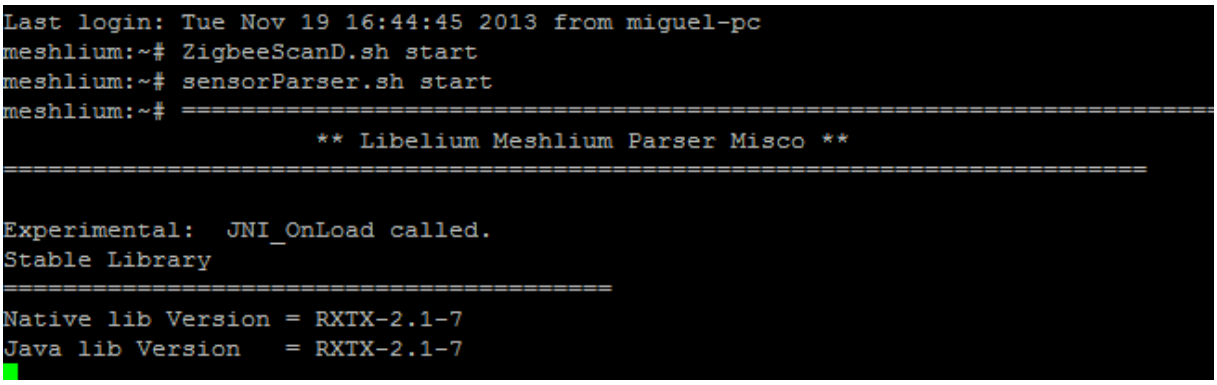



Fig. 3.69 Parseador activo

Comprobamos en el Meshlium, si la inicialización ha surtido efecto. Si *Sensor Parse Available*, el equipo está listo para recibir y administrar las tramas (Fig. 3.70).

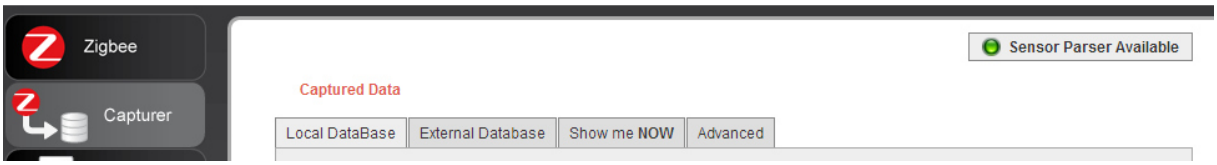



Fig. 3.70 En la pantalla de captura de datos aparece activo el parseador

## 3.5.Encendido de la Waspmote y recepción de datos en el Meshlium

Antes de activar la Waspmote, la conectamos mediante USB a un PC (Fig. 3.71).

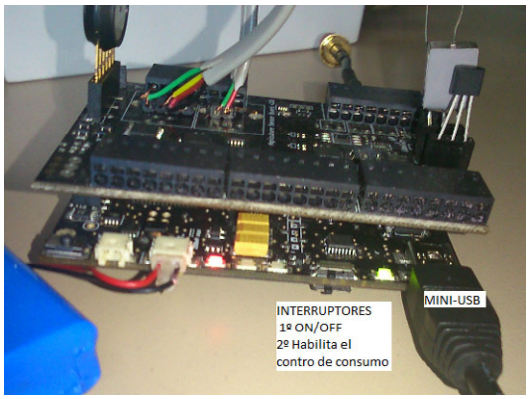



Fig. 3.71 Waspmote encendiéndose

Hay dos interruptores, el primero enciende la mota y el segundo habilita el control de consumo. Al encender la mota veremos parpadean unos led.

Accedemos al Waspmote IDE (Fig. 3.72), abrimos el archivo con el código (ver diagrama de flujo), cargamos el sketch en el dispositivo y seleccionamos *Monitor serial*, se abrirá una ventana y inicializara la ejecución. Mientras se ejecute el programa, el monitor nos servirá para saber que hace la mota en todo momento, imprimiendo los resultados, útil para depuración.

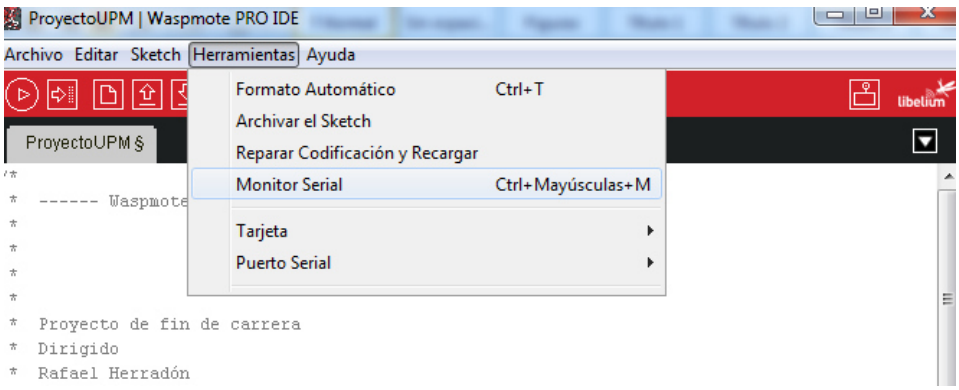



Fig. 3.72 Activando el monitor serial en el entorno de desarrollo.

La ejecución comienza con la Waspmote buscando donde está su coordinador, el Meshlium, para ello irá haciendo peticiones de conexión a una dirección concreta por cada canal radio hasta que el Meshlium la reconozca como suya y la envié los parámetros de configuración (Fig. 3.73).

```
Comenzamos...

Joined a network!
operating 16-b PAN ID: 55A6
operating 64-b PAN ID: A7F63EFD4FA10C83
channel: 14
ok
```

Fig. 3.73 Monito Serial: Conexión establecida

La Waspmote una vez conectada comienza la lectura

A continuación mide valores de los sensores como hemos explicado() temperatura, humedad, presión atmosférica y luminosidad, también incluye el valor de la batería y los envía. Vemos que la longitud de esta primera trama es de 70bytes.

```
Measuring sensors...
Temperature: 26.1290340423Â°C
Humidity: 26.3961105346%RH
Pressure: 102.3423690795kPa
Luminosity: 2.3064515590V
============================
Current ASCII Frame:
Length:  70
Frame Type (decimal): 128
HEX:     3C 3D 3E 80 05 23 33 35 36 38 39 35 37 36 39 23 4D 6F 74 61 23 31 23 54 43 41 3A 32 36 2E 31 32
String:  <=>  #356895769#Mota#1#TCA:26.12#HUMA:26.3#PA:102.34#LUM:2.306#BAT:63#
============----------------
```

**Fig. 3.74 Monitor Serial: Visualización de la primera trama**

La segunda trama contiene los valores del pluviómetro, anemómetro y dirección del viento

```
Pluviometer: 0.0000000000mm
Anemometer: 0.0000000000km/h
Vane: S
==============================
Current ASCII Frame:
Length:  47
Frame Type (decimal): 128
HEX:     3C 3D 3E 80 03 23 33 35 36 38 39 35 37 36 39 23 4D 6F 74 61 23 32 2
String:  <=>  #356895769#Mota#2#ANE:0.00#PLV:0.00#WV:78#
==============================
```

**Fig. 3.75 Monitor Serial: Visualización de la segunda trama**

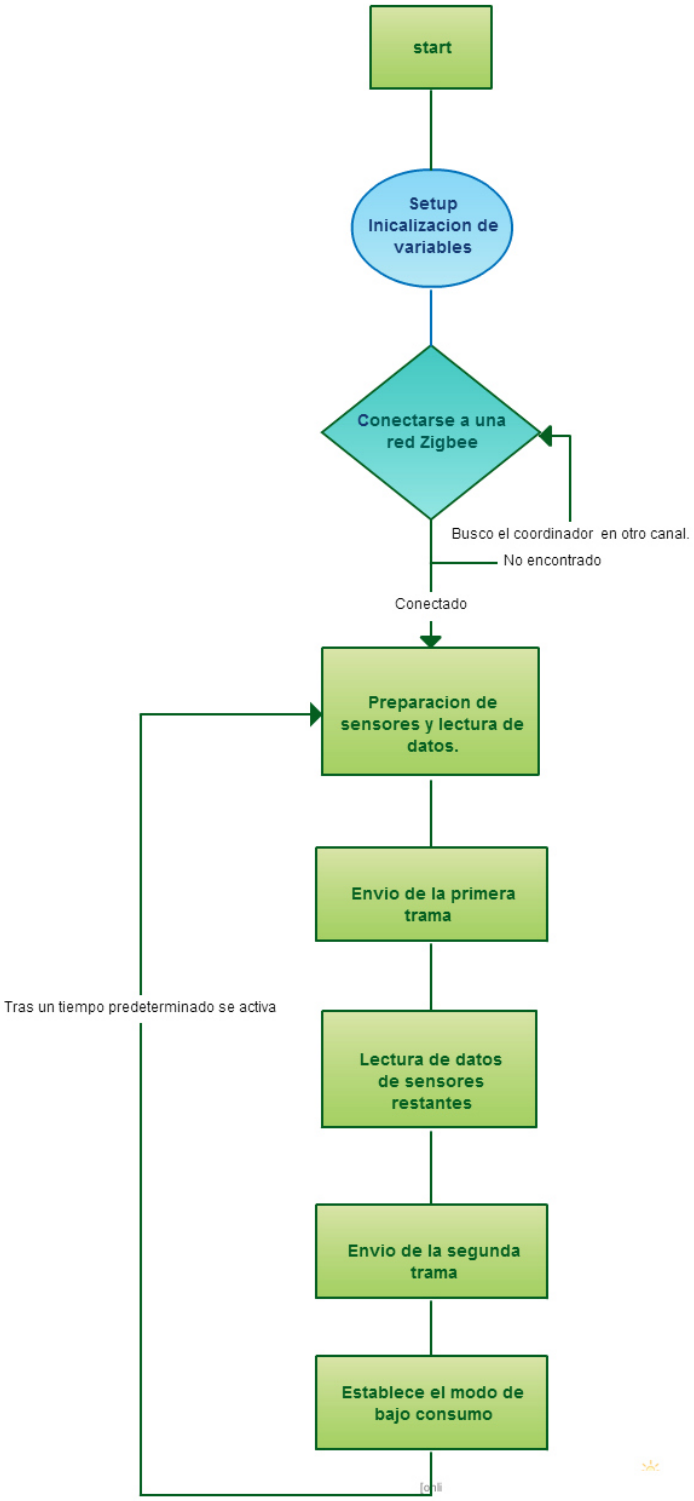
Fin del bucle, su estado cambiara a bajo consumo, y comenzara una nueva lectura de sensores. Nuestro código está programado en modo testeo, eso quiero decir que solo desactiva el modulo Xbee y vuelve otra vez a ejecutarse todo el proceso.

```
sleep mode:0
Going to sleep...
```

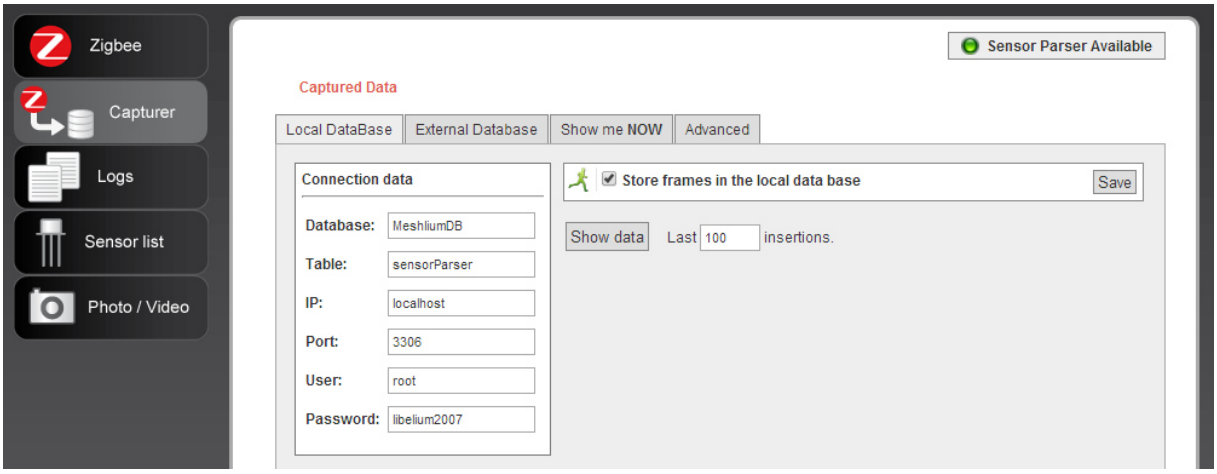
**Fig. 3.76 Monitor Serial: La Waspmote cambia a modo de bajo consumo**

## Diagrama de flujo de la Waspmote

En el siguiente diagrama se representan el conjunto de etapas anteriormente detalladas.



Mientras tanto en el Meshlium en la sección *Capturer Data* (Fig. 3.77) podemos ver en tiempo real la llegada de las tramas y como este las reconoce y las muestra divida por etiquetas. Si no fijamos en el *Frame Number* vemos que la primera en llegar, la 39 contiene 5 datos, mientras que en la trama 40 aparecen 3 elementos. (Pluviómetro (PLV) y anemómetro (ANE) con *Value* igual a cero ya que por seguridad el test se hizo dentro del laboratorio, donde se prohíbe el uso de líquidos y no hay corriente de aire, la veleta devuelve un valor numérico).




| ID | Date | Sync | ID Wasp | ID Secret | Frame Type | Frame Num | Sensor | Value |
|---|---|---|---|---|---|---|---|---|
| 9811 | 2014-02-20 17:26:51 | 0 | Mota | 356895769 | 253 | 40 | WV | 78 |
| 9810 | 2014-02-20 17:26:51 | 0 | Mota | 356895769 | 253 | 40 | PLV | 0.00 |
| 9809 | 2014-02-20 17:26:51 | 0 | Mota | 356895769 | 253 | 40 | ANE | 0.00 |
| 9808 | 2014-02-20 17:26:39 | 0 | Mota | 356895769 | 253 | 39 | BAT | 69 |
| 9807 | 2014-02-20 17:26:39 | 0 | Mota | 356895769 | 253 | 39 | LUM | 2.138 |
| 9806 | 2014-02-20 17:26:39 | 0 | Mota | 356895769 | 253 | 39 | PA | 102.09 |
| 9805 | 2014-02-20 17:26:39 | 0 | Mota | 356895769 | 253 | 39 | HUMA | 25.5 |
| 9804 | 2014-02-20 17:26:39 | 0 | Mota | 356895769 | 253 | 39 | TCA | 25.80 |

**Fig. 3.77 Capturer: Tramas recibidas por el Meshlium**

No solo se almacenan los valores para los sensores en la base de datos local, además se asigna un *ID* por dato recibido, su fecha y el *ID wasp* para saber el dispositivo que envió la información, más útil cuando tenemos varias Waspmote comunicándose a la vez.

Toda esta información puede ser eliminada accediendo a la pestaña *Advanced.* (Fig. 3.77)

Montaje final

Antes de buscar un lugar fijo donde situar nuestra Waspmote se estuvo testeando en el laboratorio, dejándola encendida durante días para comprobar que en el caso de encapsular el dispositivo herméticamente para poder tenerla en el exterior, todo funcionaria perfectamente.

**Fig. 3.78 Montaje completo, Meshlium, Waspmote y sensores**

# 4.ESCENARIO: Monitorización de sensores mediante una plataforma M2M

## 4.1.Introducción

Queremos enviar datos de sensores a una plataforma M2M gratuita llamada Xively, esta nos permitirá gestionar, almacenar y visualizar los datos mediante graficas.

Enviaremos los datos sobre dos tecnologías de comunicación diferentes, mediante la red de comunicaciones móviles utilizando un Arduino y un modulo (shield) que aporta tecnología GPRS y por otro lado una RaspberryPi que se comunicara mediante conexión cableada. Estos dos escenarios compartirán a nivel de aplicación el protocolo HTTP, a continuación detallamos las funciones utilizadas para enviar y recuperar datos de Xively(8).

Una vez los datos estén en la plataforma serán accesible por medio de una aplicación móvil en Android o mediante el interfaz WEB de Xively.

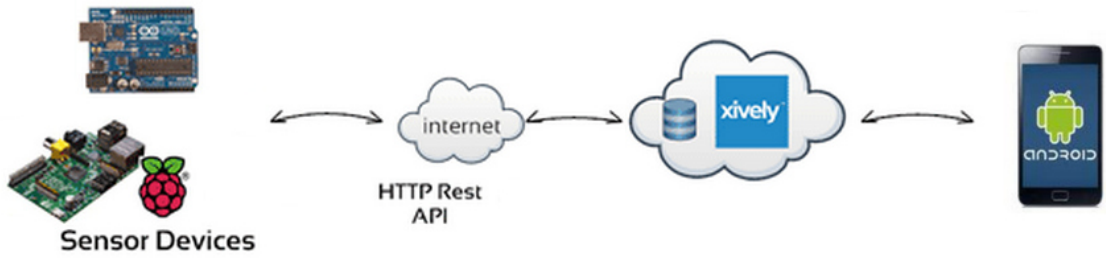



Fig. 4.1 Escenario de comunicacion

## 4.2.HTTP

Una aplicación en la nube como Xively provee al usuario de una Api REST, es decir un grupo de funciones para intercambiar datos o solicitudes de información.(9)

*REST (REST stands for Representational State Transfer)* en todos los casos, se utiliza el protocolo *HTTP*. *REST* es *un estilo de arquitectura* para el diseño de aplicaciones de red. La idea es utilizar para la conexión entre máquinas HTTP.

En muchos sentidos, la Web, está basada en HTTP, y muchos servicios se pueden ver como una arquitectura basada en *REST*.

Aplicaciones *RESTful* utilizan solicitudes *HTTP* para enviar datos (crear y/o actualizar), leer los datos (por ejemplo, realizar consultas) y eliminar datos. Por lo tanto, *REST* utiliza *HTTP* para las cuatro operaciones, *put, post, get, delete*.

Tanto las RaspberryPi como el Arduino ejecutaran PUT para el envió de datos a Xively(Fig. 4.2).

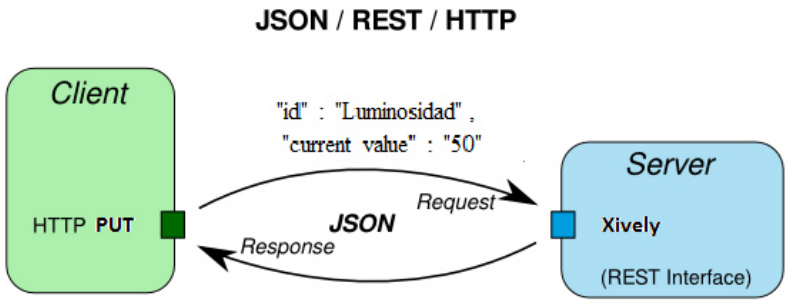



Fig. 4.2 Funcionamiento de la función PUT

Desde la aplicación móvil ejecutaremos funciones GET, es decir pediremos datos a la plataforma (Fig. 4.3 ).

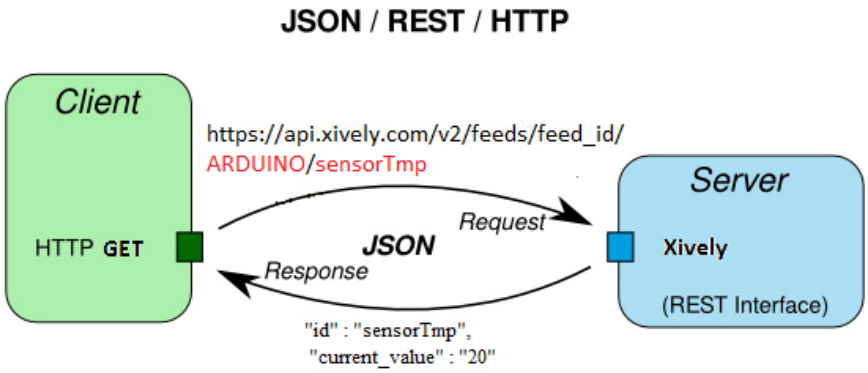


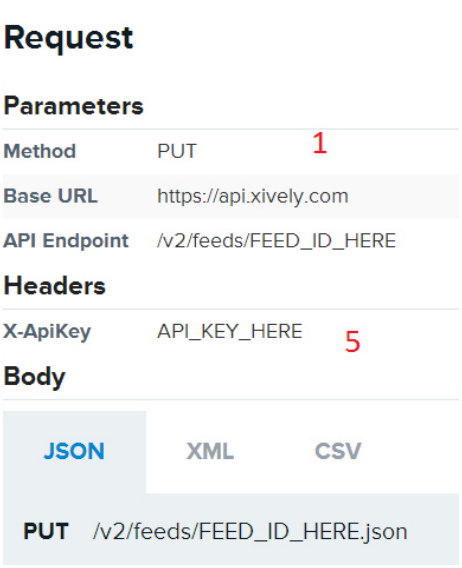

Fig. 4.3 Funcionamiento de la función GET

Nuestros dispositivos tienen que estar configurados para mandar los datos con una estructura determinada si queremos que la plataforma acepte nuestros datos. Él *API Xively* soporta tanto *HTTPS* y *HTTP*. *Hypertext Transfer Protocol Secure (HTTPS)* es método para establecer comunicaciones seguras en Internet. *HTTPS* se crea cuando el Protocolo de transferencia de hipertexto *(HTTP)* es en capas en la parte superior del protocolo *SSL/TLS* para proporcionar autenticación de punto final con el que se está comunicando, así como el cifrado de comunicaciones bidireccionales. Esto protege *HTTPS* comunicaciones de *man-in-the-middle* ataques, espionaje y manipulación.

El formato de datos *JSON* (Fig. 4.4) es especialmente adecuado para aplicaciones basadas en la web, ya que se puede analizar fácilmente usando *JavaScript* en el navegador. Sin embargo, también hace un excelente formato de serialización de datos genérico ya que tiene los gastos generales de procesamiento mucho más bajas que *XML* y usa menos ancho de banda para transmitir.

*Parameters*, *Headers* se obtienen al dar de alta el dispositivo, en los pasos 1 y 5, detallados aquí Fig. 4.22.

En la Fig. 4.4 podemos ver un ejemplo *Json*, dentro de este se aprecia la manera de enviar los datos en modo Clave-valor, el id es el *channelID* elegido para cada sensor, seguido de current_value que hace referencia al dato a almacenar.




**Request-json**

```
{
 "version" : "1.0.0" ,
  "datastreams" : [ {
"id" : "TempSensor" ,
    "current_value" : "24"
  } ,
  {
"id" : "Luminosidad" ,
"current_value" : "50"
  } ,
  {
   "id" : "sensor" ,
"current_value" : "value"
  } ]}
```

**Fig. 4.4 Izq. parámetros ofrecidos por la plataforma al dar de alta un dispositivo, Dcha. un ejemplo de envió de datos en formato Json**

## 4.3.ArduinoMega + SIM900

### 4.3.1.¿Qué es?

A continuación entramos en detalle sobre los distintos elementos que conformarán el sistema basado en la plataforma Arduino. Comenzaremos con una breve presentación de lo que es, y analizaremos los diferentes modelos de placas disponibles en el mercado. También se evaluarán las distintas shields (módulos de expansión) GSM/GPRS compatibles con esta plataforma, y los principales sensores de temperatura que tenemos a nuestra disposición.

¿Qué es Arduino? - Es una plataforma electrónica de hardware libre basada en una placa con un micro controlador. Con software y hardware flexibles y fáciles de utilizar, Arduino ha sido diseñado para adaptarse a las necesidades de todo tipo de público, desde aficionados, hasta expertos en robótica o equipos electrónicos(10). También consta de un simple, pero completo, entorno de desarrollo, que nos permite interactuar con la plataforma de manera muy sencilla. Se puede definir por tanto como una sencilla herramienta de contribución a la creación de prototipos, entornos, u objetos interactivos destinados a proyectos multidisciplinares y multitecnología. En la Fig. 4.5 podemos ver una de sus placas más vendidas a nivel mundial, se trata del modelo Arduino UNO.(11)

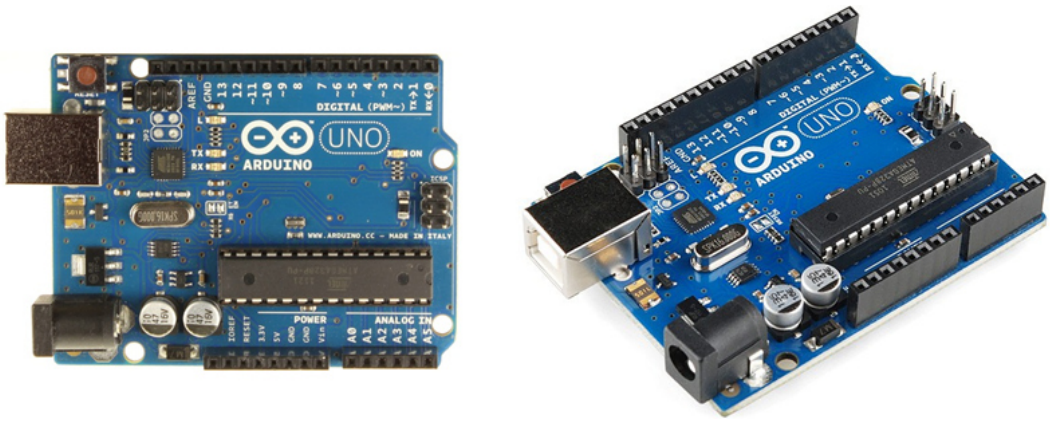



**Fig. 4.5 Fotografía de la placa Arduino UNO**

A través de Arduino podemos recopilar multitud de información del entorno sin excesiva complejidad. Gracias a sus pines de entrada, nos permite jugar con toda una gama de sensores (temperatura, luminosidad, presión, etc.) que nos brindan la capacidad de controlar o actuar sobre ciertos factores del entorno que le rodean, como por ejemplo: controlando luces, accionando motores, activando alarmas…y muchos otros actuadores.

Gracias a que la plataforma es de hardware libre, las placas Arduino pueden ser hechas a mano por cualquier aficionado o compradas ya montadas de fábrica.

Respecto al software, es totalmente gratuito, y está disponible para la descarga por cualquier interesado en la propia página web de Arduino. El entorno de desarrollo dispone de un propio lenguaje de programación para el micro controlador de la placa Arduino, basado en Processing.

"MEGA".- El Arduino Mega (Fig. 4.6) es probablemente la placa con mayores prestaciones de la familia Arduino. Cuenta con 54 pines digitales, que funcionan como entrada/salida, además de sus 16 entradas analógicas. Es la placa más grande y potente de Arduino. Es totalmente compatible con las shields Arduino UNO, y cuenta con una memoria que duplica su capacidad en comparación con el resto de placas. Arduino MEGA es por tanto la opción más adecuada para aquellos proyectos en los que se requiera un gran número de entradas y salidas disponibles, o para soportar la carga de códigos de programación pesados que no pueden almacenarse en las memorias de menor capacidad que ofrecen otras placas, como por ejemplo el modelo UNO. Quizá por este motivo, resulte interesante contar con este modelo de placa para la realización del proyecto, ya que se pretenden utilizar bastantes entradas/salidas a las que conectar la Shield GSM/GPRS y los distintos sensores/actuadores para el desarrollo del proyecto. Además, el código puede llegar a ser bastante extenso cuando intentemos centralizar todas las funciones en una sola plataforma: control de temperatura, envío y recepción de SMS, activación de alarmas, etc. Su precio ronda los 45€, aunque podemos encontrar placas no oficiales por unos 25€.

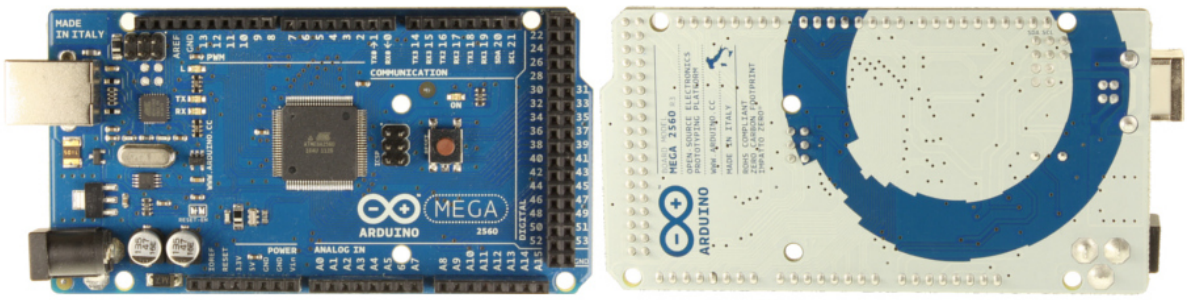



Fig. 4.6 Fotografía de la placa Arduino Mega

### 4.3.2.SIM908 GPRS/GSM

Si queremos ampliar las funcionalidades de nuestra plataforma Arduino, siempre podemos recurrir a una gran variedad de shields compatibles prácticamente con cualquiera de sus modelos. De este modo, podemos dotar al dispositivo de funciones adicionales dedicadas específicamente a ofrecer algún tipo de servicio concreto.

Un shield es un módulo de expansión en forma de placa impresa que se puede conectar a la parte superior de la placa Arduino para ampliar sus capacidades, permitiendo además

ser apiladas unas encima de otras manteniendo un diseño modular, tal como podemos ver en la Fig. 4.7.

Modulo GPRS+GPS quadband para Arduino/RaspberryPi (SIM908)

Este es el último modelo de la shield GPRS para Arduino. Gracias a que cuenta con un módulo SIM908 integrado en la propia placa, ofrece la posibilidad de utilizar la tecnología GPS para posicionamiento en tiempo real, resultando muy útil para aquellas aplicaciones en las que necesitemos conocer la ubicación de nuestro dispositivo(12).

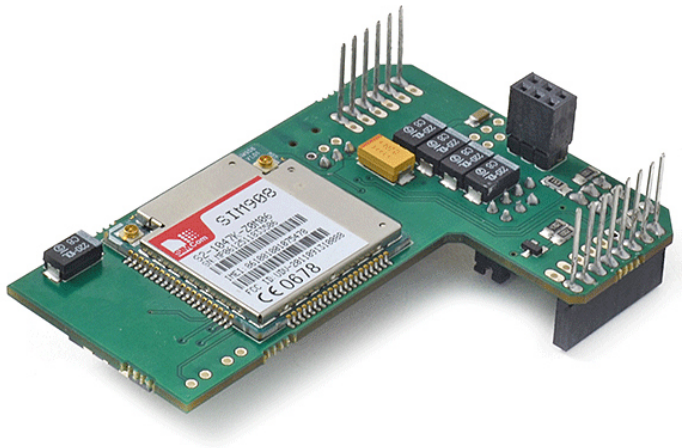


Fig. 4.7 Módulo GPRS+GPS Quadband para Arduino y RaspberryPi (SIM 908)

En nuestro caso, buscamos una shield que nos permita utilizar los sistemas de comunicaciones móviles para poder interactuar a distancia con nuestra plataforma. Navegando por internet podemos encontrar varias shields que han sido diseñadas específicamente para ofrecer servicios a través de los sistemas GSM, GPRS, 3G, o una combinación de los mismos. Además, son perfectamente compatibles con nuestra placa Arduino.

En la Fig. 4.8 podemos observar con detalle cada uno de los pines, puertos y distintos elementos que conforman esta placa:

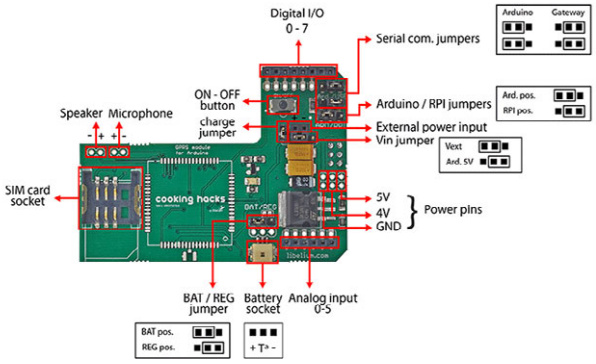



Fig. 4.8 Diagrama de conexiones del módulo GPRS+GPS Quadband (SIM908)

Junto con el módulo, se debe adquirir una antena y una fuente de alimentación externa, ya que los 5V de Arduino pueden no ser suficientes para alimentar tanto al módulo como a los componentes que conectemos a nuestra plataforma. Adquirimos también por tanto los siguientes ítems:

Antena GPRS/GSM:

Frecuencia: 900 MHz-2.1 GHz-1800 MHz
Impedancia: 50 Ohms
Polarización: vertical
Ganancia: 0 dBi
VSWR: <2:1
Potencia: 25W
Conector: UFL
Tamaño: 35mm x 6mm
Temperatura de funcionamiento: de -40°C a +85°C

Fig. 4.9 Antena GPRS

Fuente de alimentación externa 18V:

Tensión de entrada = 100-240 V
Tensión de salida = 18 VDC
Corriente máxima = 1,2 A
Diámetro del conector = 2,1 mm
Diámetro de la cubierta del conector = 5,5 mm

Fig. 4.10 Alimentacion

### 4.3.3.Montaje final

Cargamos el código en el Arduino y ensamblamos la shield GPRS junto con la antena. En modo depuración se puede alimentar el conjunto mediante USB con el fin de recibir mensajes de información en el PC.

Pulsamos durante dos segundos el botón de On de la shield y comenzara la ejecución del programa.

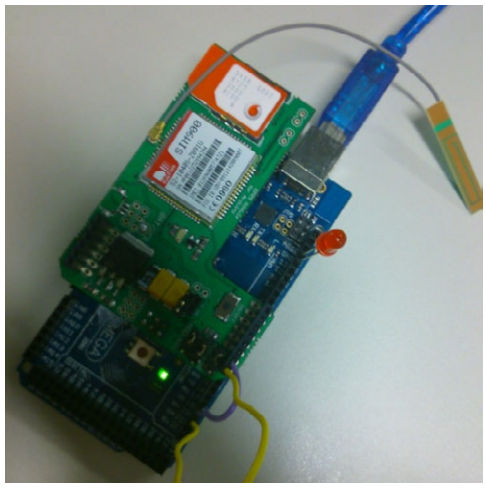


Fig. 4.11 Montaje final ArduinoMega+GPRSshield

### 4.3.4.Programación

El Arduino nos ofrece un entorno de desarrollo propio que nos permite compilar y cargar el código en el microcontrolador.

El flujo del programa comienza en la fase de *Setup,* establece que pines de entrada, salida y velocidades de comunicación que utilizara para interactuar con los sensores y la shield.

Se activa le modulo GPRS y se intenta establecer comunicación con la estación base más cercana mediante comandos AT, hasta que no recibamos la confirmación de conexión se seguirá intentando.

Una vez la conexión está establecida el Arduino lee el valor de los sensores y genera un PUT HTTP (Fig. 4.2) que envía los datos a la plataforma.

Muestra de la parte del código que se encarga de enviar los datos a Xively, en concreto aquí se envía el valor del sensor de temperatura.

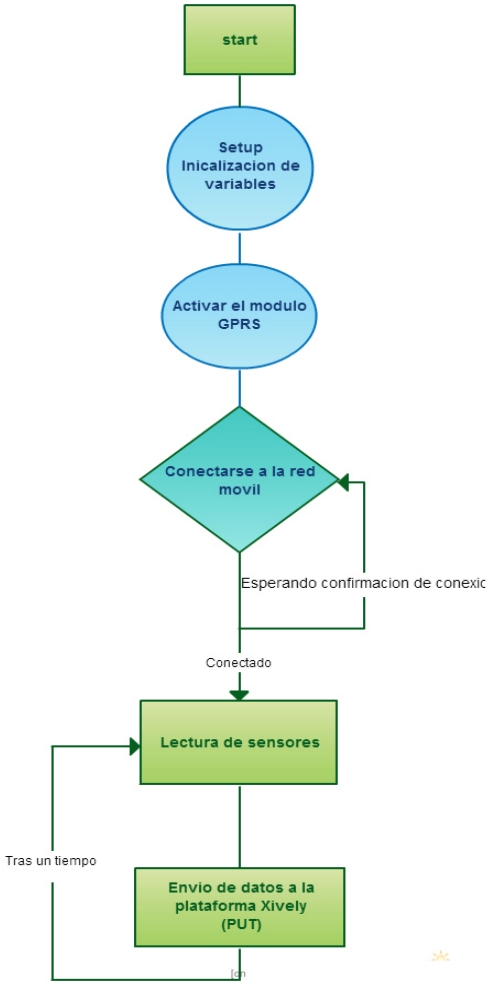



**Fig. 4.12 Diagrama de flujo del código para el Arduino**

*Serial1.print("{\"**method**\":\"**put**\",\"resource\": \"/feeds/2050485183\",\"params\": Serial1.print("\"headers\": {\"X-ApiKey\":");*
*Serial1.print("\"eaH5YmOVH0XET3ZhK9uwwj3jaGg9 XmviH3nNBbjseQ4XxN8S\"},\"body\":");*
*Serial1.print("{\"version\": "1.0.0\",\"datastreams\": ");*
*Serial1.print("[{\"**id**\":\"**sensorT**\",\"**current_value**\": **19.4**"}]}}");*

Si se quisiesen mandar varios datos o diferentes sensores basta con añadir a la última línea otros id de sensor con sus valores correspondientes. Respetar el formato que exige Xively garantiza la lectura y almacenamiento de los datos. Es interesante testear en tiempo real con la plataforma distintas formas de enviar valores.

## 4.4.RaspberryPi

### 4.4.1.¿Qué es?

Un ordenador del tamaño de una tarjeta de crédito desarrollado en el Reino Unido por la Fundación RaspberryPi(13), una iniciativa sin ánimo de lucro de un grupo de académicos de Cambridge, su utilización tenía un carácter educativo, querían acercar la informática a las escuelas. Debido a su bajo precio y filosofía de hardware libre creció su popularidad y junto con el Arduino formaron parte del un ecosistema formado por amantes de la electrónica, ingenieros y docentes que comparten en comunidad sus proyectos. Al ser un ordenador, es mucho más útil para tareas de procesamiento de datos que un Arduino, pero no tiene la facilidad de este último a la hora de conectar otros elementos.

Fig. 4.13 RaspberryPi

El primer modelo a la venta fue lanzado en febrero de 2012. Ese primer dispositivo contaba con un procesador de la familia ARM11, 256 MB de RAM, puerto HDMI, puerto Ethernet y un puerto USB con un consumo eléctrico 500 mA, (2.5 W). Es el que utilizaremos nosotros.

### 4.4.2.Configuración Hardware

Para conectar un sensor a las RaspberryPi necesitamos un MCP3008 (conversor analógico digital) actuara como un "puente" entre el modo digital y analógico. Dispone de 8 entradas analógicas y la Pi puede consultar usando 4 pines digitales. Eso hace que sea un complemento perfecto para la integración de sensores simples(14).

Para poder leer los datos analógicos que tenemos que utilizar los siguientes pines: VDD, DGND para alimentar el chip MCP3008. También necesitamos cuatro pines de datos 'SPI': DOUT(datos desde MCP3008), CLK (pin del reloj), DIN (Datos a partir de las RaspberryPi), y / CS (selector de chip).

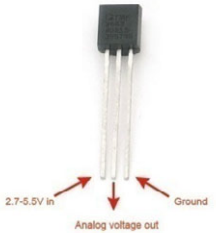



Fig. 4.14 Sensor TMP36

Por último, por supuesto, una fuente de datos analógicos, que va a utilizar el sensor de temperatura (Fig. 4.14)

El MCP3008 tiene unos cuantos pines más que necesitamos para conectar: AGND (masa analógica, que se utiliza a veces en circuitos de precisión) se conecta a GND y VREF

A continuación se muestra un diagrama de cableado. Conecte el pin 3.3V a la izquierda y el pin GND a la derecha.

MCP3008 VDD -> 3.3V
MCP3008 VREF -> 3.3V
MCP3008 AGND -> GND
MCP3008 CLK -> # 18
MCP3008 DOUT -> # 23
MCP3008 DIN -> # 24
MCP3008 CS -> # 25
MCP3008 DGND -> GND

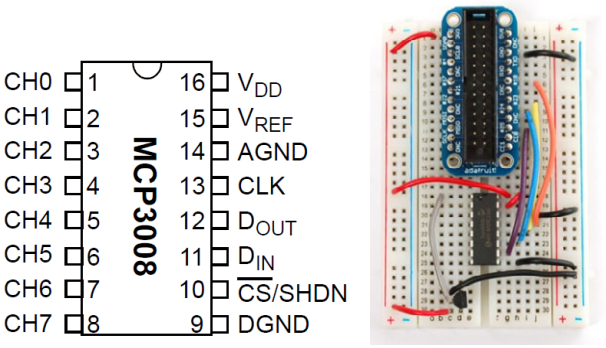



Fig. 4.15 Referencia de pines del ADC

### 4.4.3.Configuración Software

En primer lugar es necesario tener una tarjeta SD de al menos 2GB para instalar un sistema operativo, en nuestro caso instalaremos RASPBIAN(15).

Detallo los pasos a seguir para la instalación desde Ubuntu 12.04LTS por línea de comandos:

-Descargamos de la página oficial:

http://www.raspberrypi.org/downloads

-Comprobamos mediante el "hash" que es el archivo correcto

*sha1sum ~/Downloads/2014-07-04-wheezy-raspbian.zip|grep "hash"*

-Descomprimimos:

*unzip ~/Downloads/2014-07-04-wheezy-raspbian.zip*

-Insertamos la tarjeta de memoria

*df -h*

-Montamos la partición donde se encuentra nuestra SD, en mi caso es *sdb1*.

*umount /dev/sdb1*

-Escribimos mediante el commando *dd* la imagen descargada anteriormente.

*dd bs=1M if=~/Downloads/2014-07-04-wheezy-raspbian.img of=/dev/sdb1*

-Cuando termine será seguro desconectar la tarjeta del ordenador

*sudo sync*

Ahora podemos insertar la tarjeta SD en la PI, ratón, teclado por USB, la pantalla por HDMI y al conectar la alimentación. La primera vez que arranque nuestra PI, nos aparecerá la utilidad *raspi-config* con la que podremos realizar una configuración inicial.

En él podremos:

*Expand_rootfs*.- Expandir nuestra memoria SD para que ocupe toda su capacidad. Si no hiciéramos esto, por más capacidad que tuviera nuestra SD, no podríamos aprovecharlo. No requiere ninguna configuración adicional. Se elige la opción y se realiza solito.

*Overscan*.- Opción válida si vemos bordes negros en la pantalla.

*Configure_keyboard*.- Para ponerlo en español. Elegimos las siguientes opciones navegando por los menús: Configure_keyboard + Generic 105-key (intl) PC + Other + Spanish + Spanish + The default for the keyboard layout + No compose key (para el caso de España)

*Change_pass*.- Para cambiar la contraseña del usuario pi.

*Change_locate*.- es_ES.UTF-8 UTF-8 (para España)

*Change_timezone*.- Elegir Europa – Madrid (para el caso de España)

*Memory_split*.- Según el uso que le des a tu RaspberryPi puedes reservar más memoria para la CPU o la GPU (gráfica).

*overclock*.- Para forzar a tu RaspberryPi y subir los Mhz. Ten en cuenta que, como indica cuando pulsas la opción, disminuirás la vida de tu RaspberryPi y aumentarás el consumo.

*ssh*.- Para activar el acceso ssh.

*Boot_behaviour*.- Activar si quieres que el entorno gráfico aparezca por defecto (dependerá del uso que le des a tu RaspberryPi)

*Update*.- Actualizar la utilidad de configuración

Instale los paquetes de sistema:

Antes de empezar con nuestro pequeño ejemplo de la biblioteca, hay algunas tareas de instalación que se deben hacer para empezar.

En primer lugar, vamos a actualizar el software del sistema, abrimos un terminal y ejecutamos:

*sudo apt-get update*

*sudo apt- get upgrade*

Ya tendrá una versión de Python instalado por defecto, sin embargo también habrá que instalar el sistema de control de versiones Git, para descargar la última versión de nuestra biblioteca desde Github . Afortunadamente en Linux esto es tan simple como ejecutar:

*sudo apt- get install git*

Despues instalamos algunos paquetes a nivel de sistema para que nos permitirá construir nuestra aplicación de una manera limpia.

*sudo apt- get install python- setuptools*

*sudo easy_install pip*

*sudo pip install virtualenv*

Todo lo demás que vamos a instalar en este punto se instalará dentro de un *virtualenv*. El punto es crear un entorno aislado que contiene sólo las dependencias que necesitamos que no interfiera con otras aplicaciones. Así que primero vamos a crear un directorio en el que vamos a trabajar:

*Mkdir xively_tutorial*

*Xively_tutorial cd*

Vamos a crear un nuevo virtualenv en este directorio:

*virtualenv .envs/venv*

Esto crea un ambiente *python* aislado dentro de la carpeta *.envs / VENV*, pero antes de que podamos empezar a usarlo, tenemos que decirle al shell actual que este es el entorno de python que queremos usar. Hacemos esto mediante la «activación » del virtualenv :

*source .envs/venv/bin/activate*

En este mensaje se debe notar que el indicador ha cambiado, lo que indica que estamos utilizando la pitón *venv*. Ahora puede seguir adelante e instalar la biblioteca python Xively dentro de nuestra virtualenv :

*pip install xively-python*

Este comando se debe ejecutar, y debería ver que se instala la biblioteca Xively , además de las dependencias que requiere , y también se debe tener en cuenta que estas dependencias están instaladas localmente en el virtualenv .

Si se recibe un mensaje de error que dice " No se pudo encontrar una versión que cumple con el requisito ", entonces probamos con *pip install --pre xively-python*.

### 4.4.4.Montaje final

Conectaos la RaspberryPi a un monitor HDMI, un ratón y teclado. En el momento que conectemos la alimentación, comenzara a inicializarse el sistema operativo.

Abrimos un terminal, nos dirigimos a la ruta de archivos creada anteriormente y ejecutamos el script dentro del entorno configurado.

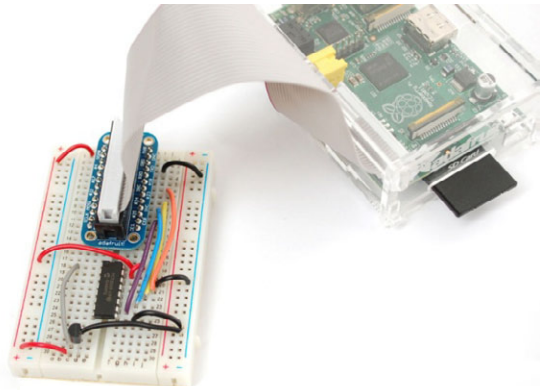


Fig. 4.16 Montaje final de la RaspberryPi

### 4.4.5.Programación

Si abrimos el fichero de código, al principio encontraremos:

```
FEED_ID="YOURFEEDID"
API_KEY = "YOURAPIKEY"
```

Estas dos variables tienen que ser sustituidas por los valores que se genera en Xively al registrar la RaspberryPi.

El script en python consiste en un bucle infinito en él que una vez leído y enviado el valor del sensor de temperatura se espera unos segundos hasta el siguiente envió.

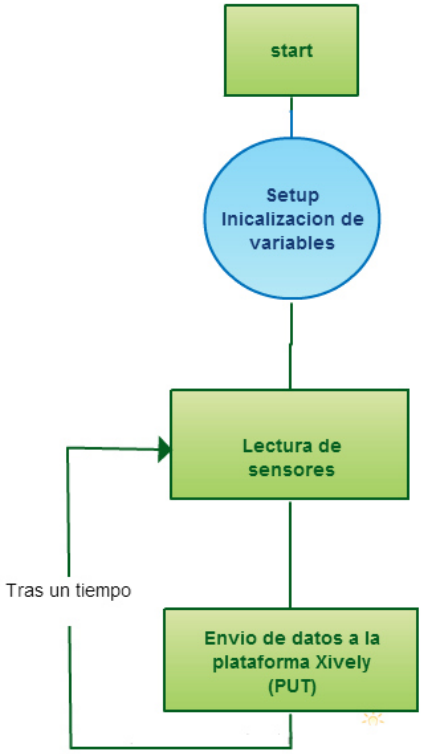



Fig. 4.17 Diagrama de flujo del código en la RaspberryPi

## 4.5.Plataforma M2M, Xively

### 4.5.1.¿Qué es Xively?

Xively es una cloud pública construida específicamente para el Internet de las Cosas, la podemos ver como una red social de sensores (flujos de datos). Los datos, los lugares y los usuarios pueden estar interconectados a través de la nube(8).

Es una plataforma M2M que nos permite conectar de manera gratuita nuestros dispositivos internet y poder gestionar, almacenar y visualizar mediante graficas la evolución de los datos.

Ofrece seguridad de extremo a extremo, ya que también estar orientada al negocio que rodea internet de las cosas. Dando soporte y ofreciendo un mayor número de funciones a empresas con fines de gestión de sus propios dispositivos o en busca de modelos que produzcan valor añadido a estos datos. En la plataforma también hay lugar para desarrolladores que quieran implementar o probar sus aplicaciones.

Xively compatible con numerosas combinaciones de software y hardware necesarios para crear sus productos y soluciones, con las bibliotecas oficiales para docenas de lenguajes y plataformas, incluyendo ObjectiveC, C, Java, JavaScript, Ruby, y mucho más. El API de conexión soporta formatos de datos JSON, XML y CSV.

Está preparada para poder gestionar 200 millones de dispositivos de 17 millones de usuarios.

### 4.5.2.Pasos de conexión de un dispositivo

Creación de una cuenta

Acceder a la página https://xively.com/ , crearnos un usuario como en cualquier otra plataforma web. Confirmación mediante email para verificar la cuenta email. Ya podemos loguearnos.

Dar de alta un dispositivo y sus distintos sensores (o flujos de datos)

En figura, podemos observar ahora que conectados como usuarios la plataforma nos ofrece en el pestaña develop en la cual podremos agregar los dispositivos y gestionarlos, en nuestro caso el ArduinoMega+SIM900 y la Raspberry PI.

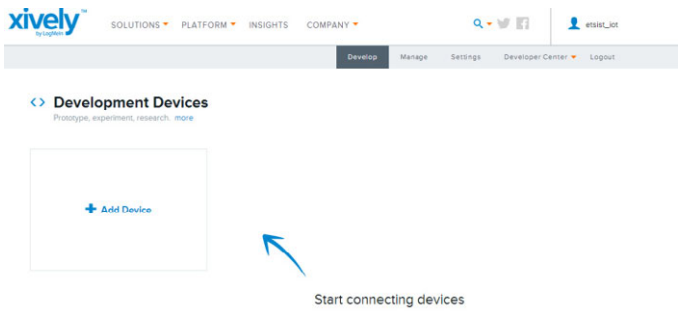



**Fig. 4.18 Ventana inicial de la plataforma**

Pulsamos en *Add Device* (Fig. 4.18) y rellenamos los campos de *Device Name* y *Device Description*, podemos además administrar el control de privacidad que queremos dar a nuestros datos. (Fig. 4.19) Seleccionando *Public Device*, compartimos con otros usuarios nos solo de Xively nuestros datos. Confirmamos y accederemos a la configuración del dispositivo. En Fig. 4.22 podemos ver la interfaz para la gestión de sensores.

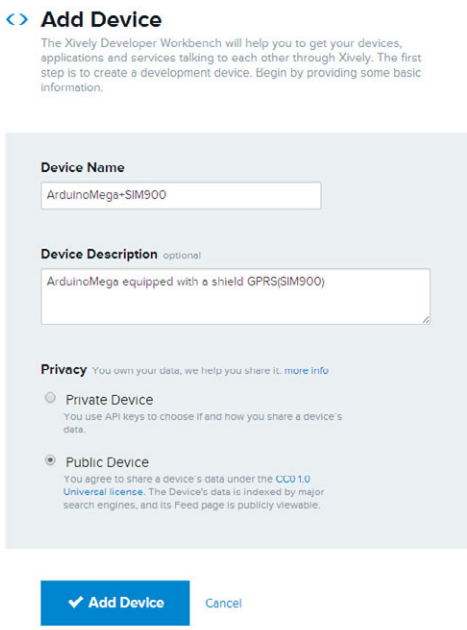



**Fig. 4.19 Alta en Xively de un dispositivo**

1.-Datos públicos de nuestro dispositivo

2.- Identificador del dispositivo, La *URL* donde podremos ver las graficas de los datos y en el caso de que sea publico nuestro dispositivo la que se compartirá. En *API EndPoint* es la dirección donde tenemos que mandar nuestros datos.

3.-Pantalla para depuración de las peticiones *HTTP*, permite ver en tiempo real lo que le llega a la plataforma desde nuestro dispositivo y si lo estamos haciendo de la manera correcta, muy útil para saber si hemos construido bien el *Json*.

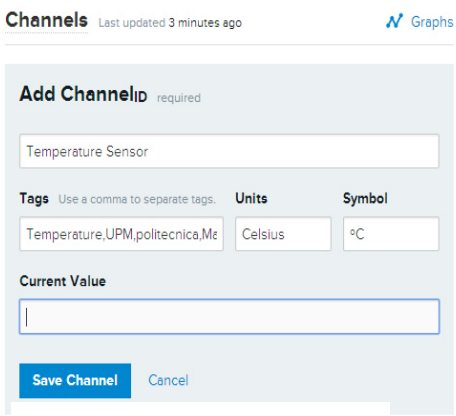



**Fig. 4.20 Añadir un sensor**

4.-*Add Channles* (Fig. 4.20) nos permite dentro de un dispositivo gestionar varios flujos de datos de sensores. Para el caso de *Arduino+SIM900* tendremos dos canales, uno para el sensor de temperatura y otro para el de luminosidad. En la figura se muestra un ejemplo de configuración de un canal, es importante recordar el nombre elegido, es decir el *ChannelID* ya que será el identificador del valor del sensor en la petición *HTTP*. La plataforma solo representa nuestros datos, no sabe lo que le estamos enviado por ello introducimos las unidades y algunos tags para que pueda ser encontrado en las opciones de búsqueda.

5.-Se genera una clave privada la cual será enviada por el dispositivo con cada dato y así la plataforma comprobara la autenticación de nuestros datos.

6.-Podemos establecer la disposición de nuestro sensores (Fig. 4.21) o actualizarla a la vez que enviamos datos, muy útil para el caso de aplicaciones con dispositivos en movimiento ayudándose de GPS o la red de móviles.

7.-Podemos administrar Alarmas cuando el dato recibido cumpla alguna condición y que nos envié el dato a una cuenta de correo o Twitter etc.

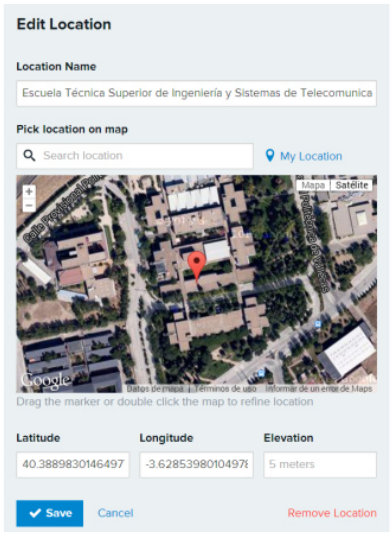



Fig. 4.21 Insertar la localización

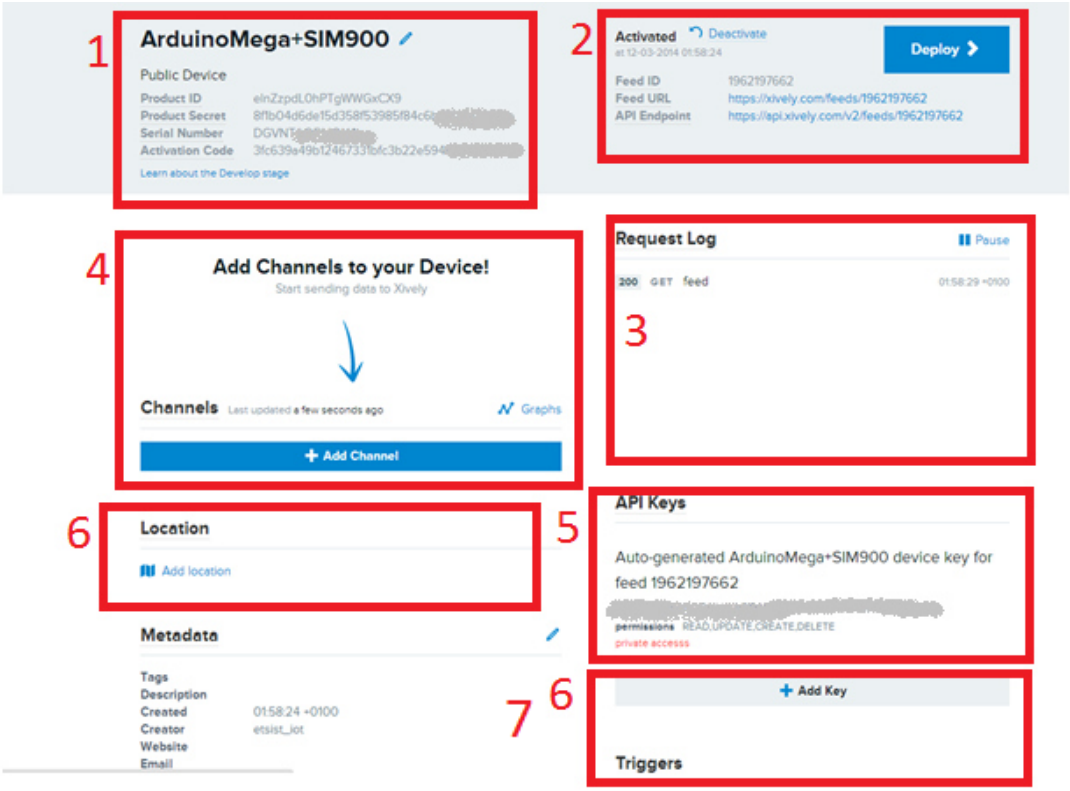



Fig. 4.22 Interfaz de gestión del sensor

### 4.5.3. Visualización de datos

Se ha creado una cuenta para la universidad y se ha dado de alta los dispositivos y algunos sensores.

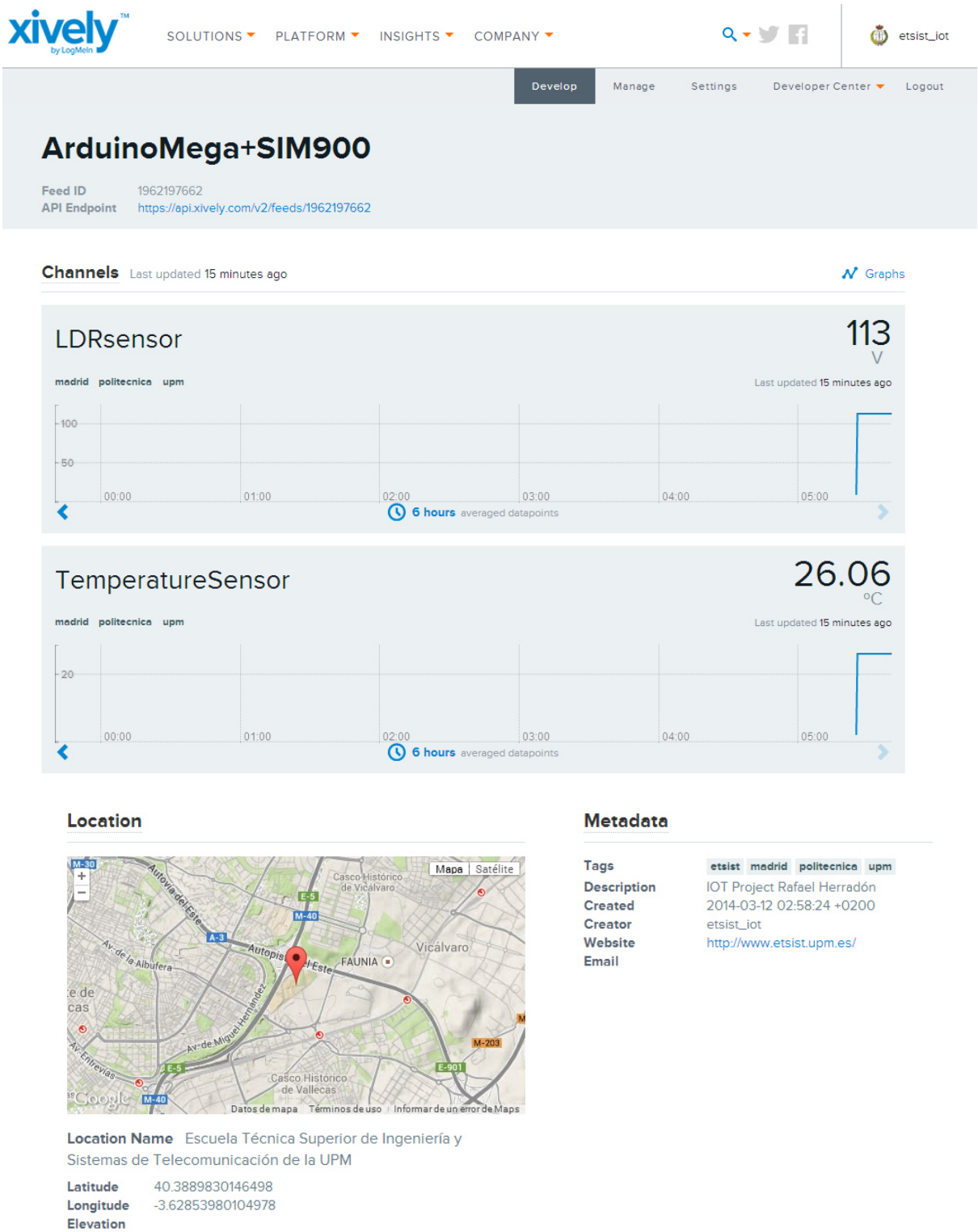



**Fig. 4.23 Visualización vía WEB de datos**

## 4.6.Aplicación Android

### 4.6.1.¿Qué es Android?

Android(16) constituye una pila de software orientada especialmente a dispositivos móviles compuesta por un sistema operativo, middleware y diversas aplicaciones de usuario. Representa la primera incursión seria de Google en el mercado móvil y nace con la pretensión de extender su filosofía a dicho sector.

Todas las aplicaciones para Android se programan en lenguaje Java, y son ejecutadas en una máquina virtual especialmente diseñada para esta plataforma. El núcleo de Android está basado en Linux 2.6.

La licencia de distribución elegida para Android ha sido Apache 2.0, lo que lo convierte en software de libre distribución. A los desarrolladores se les proporciona de forma gratuita un SDK y la opción de un plugin para el entorno de desarrollo Eclipse, que incluyen todas las API necesarias para la creación de aplicaciones, así como un emulador integrado para su ejecución.

Con Android se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wifi, Bluetooth, aplicaciones ofimáticas, videojuegos, etc.), así como poder crear aplicaciones que sean verdaderamente portables, reutilizables y de rápido desarrollo.

### 4.6.2.Objetivos de la aplicación

Hoy en día un porcentaje altísimo de la población tiene teléfono móvil, por eso un entorno de internet de las cosas donde los datos pueden ser consultados en tiempo real debe de poder gestionarse por el usuario medio de manera sencilla e instantánea.

La aplicación debe de ser la abstracción de toda la complejidad de los dispositivos y redes de conmutación con el fin de que el usuario final sea capaz de administrar sus dispositivos de una manera fácil. El desarrollar una aplicación nos permitirá reconocer las distintas herramientas que Android provee a los desarrolladores. El usuario tiene que ser capaz de introducir nuevos dispositivos y consultar datos sobre los distintos sensores. Utilizar las funciones que nos ofrece Xively para obtener los datos almacenados y módulos de Google como GoogleMaps para representar datos de localización.

### 4.6.3.Diseño

Cuando comencé a programar la aplicación uno de los problemas inesperados fue el enfrentarme a situaciones de usabilidad. Desarrollar una aplicación, sencilla y práctica, ese

fue el principal objetivo. Ponernos en la situación de un usuario sin conocimientos técnicos que necesita consultar datos de sensores y actuar en consecuencia. Su aspecto y la manera de moverse el usuario por ella tenía que ser intuitiva.

Para almacenar los datos de los sensores y sus respectivos parámetros de configuración se ha implementado en la aplicación una base de datos. Android utiliza *SQLite*, es una base de datos de código abierto, es compatible con funciones de bases de datos relacionales estándar, usa sintaxis SQL, transacciones y declaraciones preparadas. La base de datos requiere poca memoria en tiempo de ejecución (aprox. 250 KByte) que hace que sea un buen candidato de ser embebido en otros tiempos de ejecución.

Para mostrar la localización de los sensores utilizaremos las librerías de GoogleMaps para Android, para poder utilizar este API necesitamos registrarnos y obtener una clave de usuario que nos permite utilizar de manera gratuita los recursos de Google.

### 4.6.4.Desarrollo de aplicaciones en Android

Lo primero será saber que para programar aplicaciones nativas en Android, deberemos aprender a programar en el lenguaje Java, conociendo la programación orientada a objetos.

Preparar nuestro entorno de desarrollo, descargando el IDE para este caso Eclipse y instalando *plugin ADT*.(17)

*http://developer.android.com/sdk/index.html*

Una vez abrimos nuestro entorno de desarrollo, podemos descargarnos todas las versiones de Android si queremos, así como otros paquetes extra, para ello utilizaremos el Android SDK Manager. Por otro lado, podremos crear tantos emuladores de dispositivos Android como queramos: con distintos tamaños de pantalla y distintas versiones de Android. Para ello, debemos utilizar el *Android Virtual Device Manager (ADB),* podemos elegir modelos y marcas de móviles (también tablets) en concreto para simular nuestra aplicación. Otra manera de depuración, seria conectando un móvil directamente mediante microUSB/USB al PC, la carga y el seguimiento de la ejecución de aplicación se puede hacer en tiempo real lo que ayuda mucho al desarrollador a la hora de buscar y eliminar fallas del programa.

Google proporciona una documentación muy completa de todas sus herramientas y funciones.

*https://developers.google.com/android/*

### 4.6.5.Modo de funcionamiento

La aplicación se presenta mediante una pantalla de inicio (Fig. 4.26) en la que podemos añadir un nuevo dispositivo o buscar entre elementos dados de alta anteriormente. Si elegimos introducir un sensor (Fig. 4.25), accedemos al alta de dispositivos, está desarrollada para aceptar dos configuraciones, por un lado ingresando el número de teléfono, que sería la opción para la gestión de dispositivos equipados con GPRS/3G y otra opción introduciendo las claves de Xively que se generar al dar de alta un flujo de datos en la plataforma, nos permiten hacer peticiones a datos previamente almacenados.

En la opción de buscar dispositivos encontramos una lista (Fig. 4.24) de todos los elementos que previamente he tenido que añadir como si de una agenda de contactos se tratara. Esta representación acerca al usuario a los objetos conectados y encaja en el concepto de IoT.

Tras seleccionar un sensor en concreto, se salta a una pantalla donde podremos administrar (Fig. 4.28) nuestro sensor, modificar, consultar o eliminar además nos sitúa en un mapa la localización previamente almacenada. Cuando rellenamos los datos de alta de un flujo de datos en Xively se nos ofrece la opción de de posicionar en un mapa el dispositivo o introducir sus coordenadas. En la aplicación Android una vez seleccionado un dispositivo de la lista se ejecuta una petición de GET que obtiene la posición y nos sitúa el dispositivo en el mapa.

Pulsando en consultar entramos en una pantalla encargada de llevar a cabo una petición GET a Xively (Fig. 4.27) que obtiene el último dato introducido. Nos muestra el valor recibido con la hora y fecha de envió. La opción de modificar esta implementada para aquellos casos que se quiera actuar en consecuencia una vez se haya recibido un valor y el dispositivo de origen este preparado para ello.

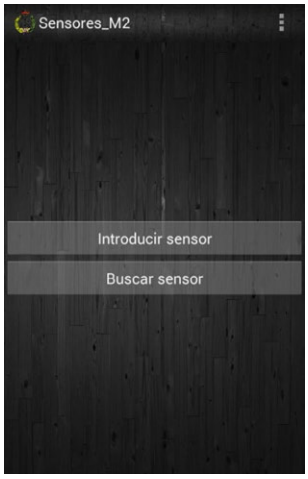



Fig. 4.26 Pantalla de inicio del APP de Android


Fig. 4.25 Pantalla de Alta de dispositivos

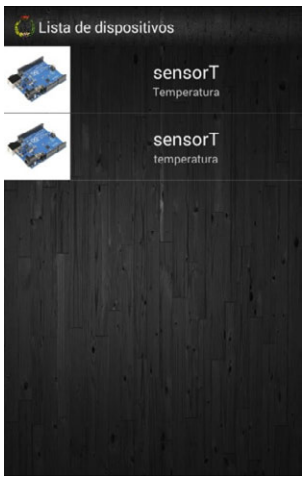



Fig. 4.24 Agenda de dispositivos

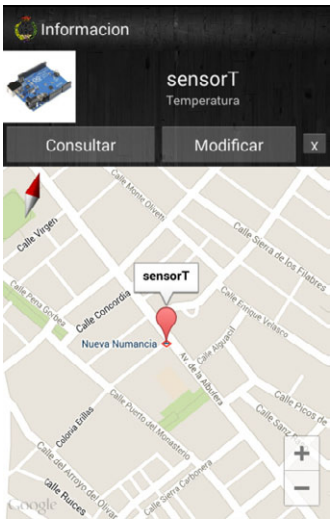



Fig. 4.28 Pantalla de Administración


Fig. 4.27 Consultar un sensor a Xively

A continuación se explica un ejemplo de comunicación entre nuestra aplicación y Xively, detallamos lo que ocurre en el momento de activar el botón "Consultar" en la Fig. 4.27. Una vez que pulsamos, nuestra aplicación genera una petición GET HTTP (Fig. 4.29), con el *feed* (identificador único para cada dispositivo que se obtiene en Xively) y nombre del sensor en nuestro caso sensorT.


Fig. 4.29 Petición HTTP para un sensor concreto

Esa petición genera una respuesta por parte de Xively que nos responde con el siguiente Json (Fig. 4.30), con el valor del último dato enviado desde el Arduino por GPRS además de otros parámetros como los valores máximo y mínimo o las unidades de medida del sensor.

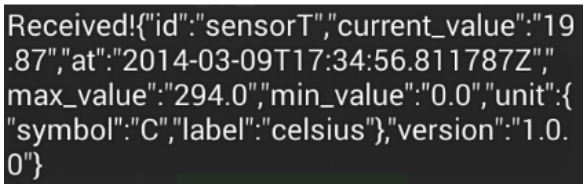



Fig. 4.30 Respuesta a nuestra petición

## 5.Mejoras futuras del proyecto

Como se ha podido entender tras la lectura proyecto, este se compone de muchos elementos distintos, nuestro objetivo era experimentar a distintos niveles de abstracción con las opciones que había en el mercado, los resultados han sido satisfactorios pero para aplicaciones más robustas habría que plantearse algunas modificaciones que para nuestro caso experimental se han descartado.

Primer escenario:

-Una mejora que tendríamos que hacer nosotros es crear un script que transmita los datos almacenados en el Meshlium a una plataforma M2M. Para los últimos modelos de Meshlium Libelium si implementa esta solución pero nuestro modelo se queda sin este tipo de servicio así que nosotros mismos somos los que deberíamos parsear los datos y enviarlos a Xively u otra plataforma.

-Si tuviéramos mas Waspmote podríamos elegir otro tipo de topología, incluso utilizar una como un router intermedio entre otra Waspmote y un Meshlium.

-Testear los modos que tiene la mota para administrar energía.

Segundo escenario:

-Establecidas unas bases iníciales en este proyecto, desde mi punto de vista se podrían hacer muchas mejoras a la aplicación Android, no solo mejorar el estilo y diseño sino gestionar mejor la participación del usuario pudiendo exigir otro tipo de interacciones con los sensores, no solo por medio de un servido como Xively si no directamente en escenarios más concretos orientados a la domótica.

-Al Arduino se le puede añadir una cámara y que nos envié fotos cada vez que el sensor de presencia detecte un valor positivo. Si tuviéramos una shield 3G podríamos recibir video de calidad en tiempo real.

-Enviar datos por GPRS desde el Arduino al Meshlium, ya que disponemos de un interfaz para ello.

## 6.Conclusiones

Para el primer sistema propuesto, la estación meteorológica, me he encontrado con dos equipos el Meshlium y Waspmote, son dos productos que forman parte del mismo ecosistema y como tales su configuración debería de ser sencilla ya que la documentación y ejemplos aportados por Libelium es extensa y suficiente para hacerlo funcionar y de esta manera lo entiende el cliente cuando los adquiere. Si se quiere desarrollar y desplegar una red con una determinada topología, jerarquía de dispositivos y que realmente su funcionamiento sea óptimo el usuario debería tener o adquirir conocimientos en redes de sensores, en este caso Zigbee. La configuración del modulo de comunicaciones de la Waspmote, Xbee no es sencilla y requiere conocer ciertos parámetros para cada topología. El proceso de conexión por parte de la Waspmote una vez desplegada la red Zigbee por el Meshlium en parte consistió en ensayo y error hasta entender correctamente el intercambio de información entre los elementos para que ambos se reconozcan y comience la transmisión de información.

La programación de la Waspmote es un proceso delicado y que requiere unos conocimientos en programación en C que permitan gestionar la memoria de una manera adecuada. Las decisiones de cómo crear una trama Zigbee influirán en el consumo de la mota, por eso en nuestro caso preferimos enviar datos sin encriptar con el fin de solo necesitar dos tramas para el envió de todos los valores de los sensores.

El Meshlium es un equipo caro y delicado y es necesario entender correctamente los protocolos de encendido y apagado con el fin de no dañar el sistema de arranque dado que es un ordenador gestionado bajo Linux. El tener que acceder al sistema mediante ssh para arrancar los script de parseo de tramas me pareció un error de diseño, debido a las advertencias en los manuales que ese tipo de gestiones por parte del usuario no las cubre la garantía. Por otra parte creo que la visualización de los datos es pobre y es necesaria una interfaz intermedia que permita observar variaciones y administrar los flujos de datos de una manera más rápida. Se intento conectar el Meshlium también a la plataforma Xively, pero para nuestro modelo Libelium no ofrecía la actualización de software necesaria. No hubiera hecho falta un equipo tan potente, hubiera bastado solo con otro modulo Xbee conectado a un ordenador normal mediante un conector USB, ya que el Meshlium es capaz de gestionar más de un elemento incluso comunicarse con otros como el.

En el segundo escenario, la parte de funcionamiento de la plataforma Xively y la comunicación mediante el API rest de HTTP me resulto sencilla hasta el momento de implementar las funciones para cada dispositivo. El Arduino con GPRS necesita de conocimientos en comandos AT, estos se encargan de gestionar la conexión con la estación base y deben de seguir una determinada secuencia si no la estación móvil deniega la conexión.

La RaspberryPi al ser un dispositivo muy popular aun teniendo que conocer el lenguaje de programación python, con ayuda de tutoriales no presento tantos problemas.

El balance para este caso es que aunque estos dispositivos no se comercialicen con fines profesionales para la industria y estén más orientados para soluciones personales o creación de prototipos cumplen su propósito y gracias a la comunidad de Arduino y RaspberryPi permiten reducir la curva de aprendizaje.

El desarrollar una aplicación Android desde cero me permitió explorar la visión personal que tengo del internet de las cosas y como me gustaría que los usuarios interactuaran con los objetos, dejando de lado la complejidad técnica, que la hubo, el diseñar una interfaz de usuario útil y abstrayendo al usuario de requerimientos técnicos supuso un verdadero reto.

El trabajar con software y hardware libre permite a los desarrolladores despreocuparse de problemas de licencias y patentes. Puedes crear tus propios dispositivos y soluciones de una manera económica, compartiendo conocimiento, ayudando a otros usuarios y disfrutando en el proceso.

## 7.Anexo A: Referencias


1. **Libelium.** http://www.libelium.com/development/waspmote.

2. **Libelium.** http://www.libelium.com/development/meshlium.

3. **Gislason, Drew.** *Zigbee Wireless Networking.* s.l. : Elsevier Science.

4. **Faludi, Robert.** *Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing.* s.l. : O'Reilly Media.

5. **Digi.** http://www.digi.com/xbee/.

6.**Libelium.** http://www.libelium.com/development/waspmote/documentation/agriculture-board-technical-guide/.

7. **Libelium** http://www.libelium.com/development/waspmote/documentation/ide-guide/

8. **Xively.** https://xively.com/.

9. Explicacion y ejemplos HTTP. http://code.tutsplus.com/tutorials/a-beginners-guide-to-http-and-rest--net-16340.

10. http://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento/.

11. **Arduino.** http://www.arduino.cc/es/.

12.**CookingHacks.**http://www.cooking-hacks.com/index.php/documentation/tutorials/arduino-gprs-gsm-quadband-sim900.

13. **RaspberryPi.** http://www.raspberrypi.org/.

14. **Adafruit.** http://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor.

15. **RaspberryPi.** http://www.raspberrypi.org/downloads/.

16. **Android.** http://www.android.com/.

17. **Eclipse.** http://developer.android.com/sdk/index.html.

18. http://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento/.